MCQ Tests 1-18 with Keys

Technical - Programming Logic Using C

(TCS NQT Ninja, DXC)

1. Neelam wants to share her code with a colleague, who may modify it. Thus she wants to include the date of the program creation, the author and other with the program. What component should she use? [C]

A. Header
B. Iteration
C. Comments
D. Pre processor

2. Which is used to convert source code to target language [C]

A. Loader
B. Linker
C. Compiler
D. Executor

3. A 10-bit unsigned integer has the following range: [C]

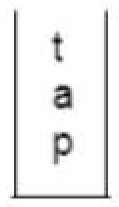
A. 0-1000
B. 0-1024
C. 0-1023
D. 0-1025

Answer: Signed means the leftmost bit of binary representation of that integer is used to show the sign of the integer. 1 means negative and 0 means positive. And rest of the bits represent the magnitude of the integer.

Unsigned means all the bits in the binary representation of the integer are used to show its magnitude and hence it can take only a positive value.

As yours question we have a 10 bit unsigned integer, hence all the 10 bits will represent the magnitude of the integer. So the range will go from (0000000000) to (1111111111)which in decimal means 0 to 2^10−1 i.e 0to 1023.

4. What will be the result if the given stack is popped? [B]



a) pat **b) tap** c) atp d) apt

Answer: b

Explanation: The word ‘pat’ is pushed onto the stack. When the characters of the stack are popped one by one, the word ‘tap’ is obtained.

5. Tricha wants to store a list of binary data. Which of the following data types should she use? [A]

A. **Boolean**
B. Integer
C. Char
D. Float

Explanation: As boolean can store only two values(i.e, 1 and 0), so Tricha can store a list of binary data in boolean datatype.

6. Which of the following options is an exception to being a part of composite data types? [D]

A. Union
B. Structure
C. Array
D. Stack

Stack is an Application of Arrays. It is not a data type

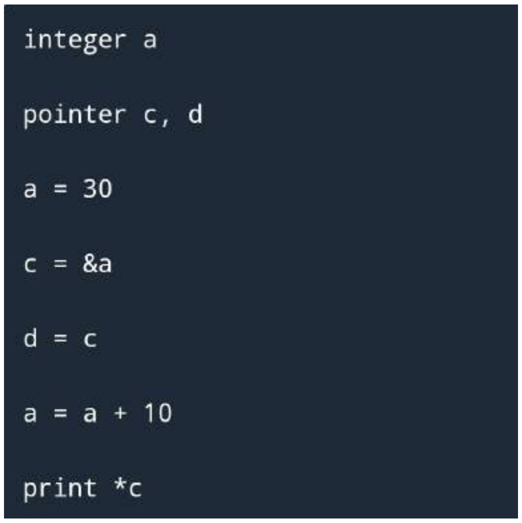
7. A data type is stored as a 6 bit signed integer. Which of the following cannot be represented by this data type? [C]

A. -12
B. 6
C. 18

For n bits, the range of 2’s complement signed integers that are allowed is -2^(n-1) to (2^(n-1))-1.

For the given question, we asked the range of 6 bit signed integers, so for 6 bits, the range of 2’s complement signed integers that are allowed is -2^(5) to (2^(5))-1. The answer is -32 to +31 both numbers included. Hence, 32 and 64 both cannot be represented by using 6 bits.

8. What is the output of the following code statements? The compiler saves the first integer at the memory location 4165 and the rest at consecutive memory spaces in order of declaration. Integer is one byte long. [B]



```
integer a

pointer c, d

a = 30

c = &a

d = c

a = a + 10

print *c
```

A. 30
B. 40
C. 4165
D. 4166

a is an integer variable and c, d are pointer variable.

Initially, a is assigned with 30. And, c is pointed to a, then d is assigned with c, so c and d both points a, a is incremented by 10 so a will become 40, So, *c will print the value of the variable c pointed 40.

9. A language has 28 different letters in total. Each word in the language is composed of a maximum of 7 letters. You want to create a data type to store a word of this language. You decide to store the word as an array of letters. How many bytes will you assign to the data type to be able to store all kinds of words of the language? [A]

A. 7
B. 75
C. 29
D. 126

The size of char data type is 1. In this question, we need to store the name of 7 letters in a char array. So, to store 7 letters we need a char array of 7 size. Hence, 7 bytes are required.

10. Parul takes as input two numbers: a and b. a and b can take integer values between 0 and 255. She stores a, b and c as 1-byte data type. She writes the following code statement to process a and b and put the result in c.
c = a + 2*b
To her surprise, her program gives the right output with some input values of a and b, while giving an erroneous answer for others. For which of the following inputs will it give a wrong answer? [B]

A. a=200, b=10
B. a=10, b=200
C. a=50, b=100
D. a=100, b=50

Answer:
0 to 255 integers are accepted

when a=10 b=200
c =410 > 255

when a=200 b=10
c=220<255

when a=50 b=100
c=250<255

when a=100 b=50
c=200<255

so, when a=10 and b=200 will give wrong output

11. A data type is stored as a 6 bit signed integer. Which of the following cannot be represented by this data type?

A. -12
B. 5
C. 25
D. 32

Ans : D
Explanation: 2^6 = 64. So you can represent 64 values. This means 0 to 63 for unsigned integer and -32 to 31 for signed integer (twos complement).

12. Which has the highest precision? [C]

A. float
B. unsigned long int
C. double
D. long int

Explanation: double has the highest precision

13. How can we insert different types of data into the stack. [D]

A. Not possible
B. By implementing union
C. Structure data type
D. Both B and C

Explanation: Union and structure are the special data type that allows storing different data types in the same memory

14. The scope of an automatic variable is: [D]

A. Within the block it appears
B. Within the blocks of the block it appears
C. Until the end of the program
D. Within the block it appears & Within the blocks of the block, it appears.

Explanation: Within the block, it appears and Within the blocks of the block it appears.

15. What argv mean in command line argument? [C]

A. Array of pointers
B. pointer to a character array
C. Array of character pointers
D. Array of Strings

Explanation: argv(Argument Vector) is an array of character pointers listing all the arguments. If argc is greater than zero, the array elements from argv[0] to argv[argc-1] will contain pointers to strings. Argv[0] is the name of the program, After that, till argv[argc-1] every element is the command-line argument.

Test-2 Prog Logic with solutions

1) What is the 16-bit compiler allowable range for integer constants?

A. -3.4e38 to 3.4e38
B. -32767 to 32768
C. -32668 to 32667
D. -32768 to 32767

Answer: (d) -32768 to 32767

Explanation: In a 16-bit C compiler, we have 2 bytes to store the value.

The range for signed integers is -32768 to 32767.
The range for unsigned integers is 0 to 65535.
The range for unsigned character is 0 to 255.

2) Study the following program:

```
main()
{printf("javatpoint");
main();}
```

What will be the output of this program?

A. Wrong statement
B. It will keep on printing javatpoint
C. It will Print javatpoint once
D. None of the these

Answer: (b) It will keep on printing javatpoint

Explanation: In this program, the main function will call itself again and again. Therefore, it will continue to print javatpoint.

3) What is required in each C program?

A. The program must have at least one function.
B. The program does not require any function.
C. Input data
D. Output data

Answer: (a) The program must have at least one function.

Explanation: Any C program has at least one function, and even the most trivial programs can specify additional functions. A function is a piece of code. In other words, it works like a sub-program.

4) What will this program print?

```
main()
{
  int i = 2;
  {
    int i = 4, j = 5;
     printf("%d %d", i, j);
  }
  printf("%d %d", i, j);
}
```

A. 4525
B. 2525
C. 4545
D. None of the these

Answer: (a) 4525

Explanation: In this program, it will first print the inner value of the function and then print the outer value of the function.

5) Which of the following comment is correct when a macro definition includes arguments?

A. The opening parenthesis should immediately follow the macro name.

B. There should be at least one blank between the macro name and the opening parenthesis.

C. There should be only one blank between the macro name and the opening parenthesis.

D. All the above comments are correct.

Answer: (a) The opening parenthesis should immediately follow the macro name.

6) What is a lint?

A. C compiler
B. Interactive debugger

C. Analyzing tool
D. C interpreter

Answer: (c) Analyzing tool

Explanation: Lint is an analyzing tool that analyzes the source code by suspicious constructions, stylistic errors, bugs, and flag programming errors. Lint is a compiler-like tool in which it parses the source files of C programming. It checks the syntactic accuracy of these files.

7) What is the output of this statement "printf("%d", (a++))"?

A. The value of (a + 1)
B. The current value of a
C. Error message
D. Garbage

Answer: (b) The current value of "a".

8) Study the following program:

```
main()
{
   char x [10], *ptr = x;
  scanf ("%s", x);
  change(&x[4]);
}
 change(char a[])
 {
   puts(a);
 }
```

If abcdefg is the input, the output will be

A. abcd
B. abc
C. efg
D. Garbage

Answer: (c) efg

9) Study the following program:

```
main()
```

```
{
  int a = 1, b = 2, c = 3:
  printf("%d", a + = (a + = 3, 5, a))
}
```

What will be the output of this program?

A. 6
B. 9
C. 12
D. 8

Answer: (d) 8

Explanation: It is an effect of the comma operator.

a + = (a + = 3, 5, a)

It first evaluates to "a + = 3" i.e. a = a + 3 then evaluate 5 and then evaluate "a".

Therefore, we will get the output is 4.

Then,

a + = 4

It gives 8 as the output.

10) What does this declaration mean?

int x : 4;

A. X is a four-digit integer.
B. X cannot be greater than a four-digit integer.
C. X is a four-bit integer.
D. None of the these

Answer: (c) X is a four-bit integer.

Explanation: This means, "X" is a four bit integer.

11) Why is a macro used in place of a function?

A. It reduces execution time.
B. It reduces code size.

C. It increases execution time.
D. It increases code size.

Answer: (d) It reduces code size.

Explanation: Macro is used in place of a function because it reduces code size, and very efficient.

12) In the C language, the constant is defined _______.

A. Before main
B. After main
C. Anywhere, but starting on a new line.
D. None of the these.

Answer: (c) Anywhere, but starting on a new line.

Explanation: In the C language, the constant is defined anywhere, but starting on a new line.

13) How many times will the following loop execute?

```
for(j = 1; j <= 10; j = j-1)
```

A. Forever
B. Never
C. 0
D. 1

Answer: (a) Forever

14) A pointer is a memory address. Suppose the pointer variable has p address 1000, and that p is declared to have type int*, and an int is 4 bytes long. What address is represented by expression p + 2?

A. 1002
B. 1004
C. 1006
D. 1008

Answer: (d) 1008

15) What is the result after execution of the following code if a is 10, b is 5, and c is 10?

```
If ((a > b) && (a <= c))
        a = a + 1;
else
        c = c+1;
```

A. a = 10, c = 10
B. a = 11, c = 10
C. a = 10, c = 11
D. a = 11, c = 11

Answer: (b) a = 11, c = 10

Test-3 Prog Logic With Solutions

1. Which one of the following is a loop construct that will always be executed once?

a. for

b. while

c. switch

d. do while

Answer: (d) do while

Explanation: The body of a loop is often executed at least once during the do-while loop. Once the body is performed, the condition is tested. If the condition is valid, it will execute the body of a loop; otherwise, control is transferred out of the loop.

2. Which of the following best describes the ordering of destructor calls for stack-resident objects in a routine?

a. The first object created is the first object destroyed; last created is last destroyed.

b. The first object destroyed is the last object destroyed; last created is first destroyed.

c. Objects are destroyed in the order they appear in memory, the object with the lowest memory address is destroyed first.

d. The order is undefined and may vary from compiler to compiler.

Answer: (b) The first object destroyed is the last object destroyed; last created is first destroyed.

3. How many characters can a string hold when declared as follows?

```
1. char name[20]:
```

a. 18
b. 19
c. 20
d. None of the these

Answer: (b) 20

4. Directives are translated by the

a. Pre-processor
b. Compiler
c. Linker
d. Editor

Answer: (a) Pre-processor

Explanation: In C language, the pre-processor is a macro processor that is dynamically used by the C programmer to modify the program before it is properly compiled (Before construction, pro-processor directives are implemented).

5. How many bytes does "int = D" use?

a. 0
b. 1
c. 2 or 4
d. 10

Answer: (c) 2 or 4

Explanation: The int type takes 2 or 4 bytes.

6. Which of the following will copy the null-terminated string that is in array src into array dest?

a. dest = src;

b. dest == src;

c. strcpy(dest, src);

d. strcpy(src, dest);

Answer: (c) strcpy(dest, src)

Explanation: strcpy is a string function that is used to copy the string between the two files. strcpy(destination, source)

7. What is the maximum number of characters that can be held in the string variable char address line [40]?

a. 38

b. 39

c. 40

d. 41

Answer: (b) 39

---

8. What will the result of num1 variable after execution of the following statements?

```
int j = 1, num1 = 4;
while (++j <= 10)
{
  num1++;
}
```

a. 11

b. 12

c. 13

d. 14

Answer: (c) 13

9. What will be the output of this program?

```
int main()
{
int a=10, b=20;
printf("a=%d b=%d",a,b);
a=a+b;
b=a-b;
a=a-b;
printf("a=%d b=%d",a,b);
return 0;
}
```

a. a = 20, b = 20

b. a = 10, b = 20

c. a = 20, b = 10

d. a = 10, b = 10

Answer: (c) a = 20, b = 10

Explanation: This program is a swapping program.

a = a + b → a = 10 + 20 → a = 30

b = a - b → b = 30 - 20 → B = 10

a = a - b → a = 30 - 10 → a = 20

Study the following statement

```
for (i = 3; i < 15; i + = 3)
{printf ("%d", i);
 ++i;
}
```

10. What will be the output?

a. 3 6 9 12
b. 3 6 9 12 15
c. 3 7 11
d. 3 7 11 15

Answer: (c) 3 7 11

11. Study the following statement

```
main()
{
  char *s = "Hello,"
  "World!";
  printf("%s", s);
}
```

What will be the output?

a. Hello, World!

b. Hello,
   World!

c. Hello

d. Compile error

Answer: (a) Hello, World!

Explanation: The output of this program is "Hello, World!". This program's output will not appear in the new line because the \ n escape sequence has not been used in this program.

---

12. Study the following array definition

```
int num[10] = {3, 3, 3};
```

Which of the following statement is correct?

a. num[9] is the last element of the array num

b. The value of num[8] is 3

c. The value of num[3] is 3

d. None of the above

Answer: (a) num[9] is the last element of the array num

Explanation: The num[9] is the last element of the array number because the total element in this array is 10, and the array starts with 0, so the last element of the array is the num[9]

13. What will the output after execution of the following statements?

```
void main()
{
   int i = 065, j = 65;
   printf ("%d %d", i, j);
}
```

a. 065 65

b. 53 65

c. 65 65

d. Syntax error

Answer: (b) 53 65

Explanation: This value (065) is an octal value, and it equals to the decimal value 53.

14. What will the output after execution of the following statements?

```
#include<stdio.h>void main()
{
    int  i=10;
    printf("i=%d", i);
    {
        int  i=20;
             printf("i=%d", i);
             i++;
             printf("i=%d", i);
    }
    printf("i=%d", i);}
```

A. 10 10 11 11
B. 10 20 21 21
C. 10 20 21 10
D. 10 20 21 20

Answer: Option C

Explanation - The scope of second declaration of i is limited to the block in which it is defined. Outside of the block variable is not recognized.

15. Which data type cannot be checked in switch-case statement?
A. integer
B. float
C. enum
D. character

Answer- B

Explanation:- In C-language switch/case statement is defined by the language specification to use an int value therefore we cannot use a float value in switch/case statement.

Test-4 Prog Logic With Solutions

1. Array is a ________ data structure.

a. Non-linear
b. Primary
c. Linear
d. Data type

Answer: (c) Linear

Explanation: An array is a non-primitive and linear data structure that only stores a similar data type.

2) Which of the following statement is correct about the array?

a. In the array, users can only allocate the memory at the run time.
b. In the array, users can only allocate the memory at the compile time.
c. The array is a primitive and non-linear data structure that only stores a similar data type.
d. All of the these

Answer: (b) In the array, users can only allocate the memory at the compile time.

Explanation: An array is a non-primitive and linear data structure that only stores a similar data type. In array, users can only allocate the memory at the compile time.

3. In the following program fragment, s and b are two integers:

1. b = s + b
2. s = b - s
3. b = b - s

What does it intend to do?

a. Exchange the values of s and b

b. Transfer the values of s and b

c. Transfer the values of b and s

d. Add or subtract the values of s and b

Answer: (a) Exchange the values of s and b

Explanation: The intention of this program fragment is to exchange (swap) the values of s and b. Let us take an example for better understand:

1. s = 1
2. b = 2
3. b = s + b
4. b = 1 + 2
5. b = 3
6. s = b - s
7. s = 3 - 1
8. s = 2
9. b = b - s
10. b = 3 - 2
11. b = 1

---

3. Which of the following functions is already declared in the "header file"?

a. User-define function

b. Built-in function

c. C function

d. None of the these

Answer: (b) Built-in function

Explanation: Built-in functions are those functions whose prototypes are preserved in the header file of the "C" programming language. These functions are called and executed only by typing their name in the program. For example, scanf(), printf(), strcat(), etc.

4. Which of the following keywords is used to prevent any kind of change in a variable?

a. continue

b. const

c. struct

d. extern

Answer: (b) const

Explanation: Constant is a variable whose value cannot be changed once assigned. Constant is also called literals. It can be of any basic data type such as char, integer, float, and string. It can be defined anywhere in the program but in a new line. It is represented by the const keyword.

5. Which of the following declarations is invalid in C language?

a. char *str = "Visual Studio Code is best to compile platform ";

b. char str[] = "Visual Studio Code is best to compile platform ";

c. char str[20] = "Visual Studio Code is best to compile platform ";

d. char[] str = "Visual Studio Code is best to compile platform ";

Answer: (d) char[] str = "Visual Studio Code is best compile platform ";

Explanation: This declaration is valid in java language, but not in C language. Therefore, option (d) is the correct answer.

6. The enum keyword is used to assign names to the _______ constants.

a. Integer

b. String

c. Character

d. All of the these

Answer: (a) Integer

Explanation: Enumeration is a user-defined data type in C language that is used to assign names to integral constants. It is represented by the "enum" keyword.

7. Study the following program fragment. Hint: Each alphabet has an ASCII number to it. Z's ASCII number is 90. Will it store Z or 90 in ch?

```
char ch = 'Z'
```

What will store in ch?

a. Z

b. 90

c. 91

d. 122

Answer: (b) 90

Explanation: The capital 'Z' value is 90 accordingly to the ASCII table. Therefore, 90 will be assigned to ch variable.

8. How many times "Technictiming" is printed?

```
#include<stddio.h>
int main()
{
int x;
for(x=-1; x<=10; x++)
{
if(x < 5)
continue;
else
break;
printf("Technictiming");
} return 0;
}
```

A. Infinite times
B. 0 times
C. 10 times
D. 11 times

Answer: B

Explanation:- In program the x is initialized with -1. As x < 5 (since x is -1) it will start with continue statement. Continue means "stop the current iteration and proceed to the next iteration". Therefore x becomes 0 now. This will take place until x becomes 5.

Now if the value of x=5, it will enter the else part where it encounters the break statement, as a result it will come out of for loop. Hence it will not go to printf statement. Therefore technictiming will be printed 0 times.

9. What is the output of C Program.?

```
int main()
{
while(TRUE)
{
printf("LIONS");
break;
}
return 0;
}
```

A. LIONS
B. LIONS is printed unlimited number of times.
C. No output
D. Compiler error.

Ans :- D

Explanation:- while(TRUE) or while(true) does not work. true is not a keyword.

10. What is the output of the C statement.?

```
int main()
{
   int a=0;
   a = 5<2 ? 4 : 3;
   printf("%d",a);
   return 0;
}
```

A) 4
B) 3
C) 5
D) 2

Answer B
Explanation: 5<2 is false. So 3 will be picked and assigned to the variable a.

11. What is the output of the C Program.?

```
int main()
{
   int a=0;
   a = 5>2 ? printf("4"): 3;
   printf("%d",a);
   return 0;
}
```

A) compiler error
B) 14
C) 41
D) 0

Answer C
Explanation: 5>2 is true. So expression1 i.e printf("4) is executed printing 4. Function printf() returns 1. So a value is 1.

12. What is the output of the C Program.?

```
int main()
{
   int a=0;
```

```
    a = (5>2) ? : 8;
    printf("%d",a);
    return 0;
}
```

A) 0
B) 1
C) 8
D) compiler error

Answer: B

Explanation:
expression1 = empty
expression2 = 8
If no expression is specified, it will be treated as 1.

13. Choose a statement to use C If Else statement.

A) else if is compulsory to use with if statement.
B) else is compulsory to use with if statement.
C) else or else if is optional with if statement.
D) None of the above

Answer C

14. Choose a correct C Statement using IF Conditional Statement.

A)
```
if( condition )
{
    //statements;
}
```

B)
```
if( condition )
{
    //statements;
}
else
```

```
{
   //statements;
}
```

C)

```
if( condition1 )
{
   //statements;
}
else if( condition2)
{
   //statements;
}
else
{
   //statements;
}
```

D) All the above.

Answer D

15. What is the output of the C Program.?

```
int main()
{
   if( 4 > 5 )
      printf("Hurray..\n");
      printf("Yes");

   return 0;
}
```

A) Yes
B) Hurray.. Yes
C) Hurray..Yes
D) No Output

Answer A

Explanation: To include more than one statement inside If block, use { } braces. Otherwise, only first statement after if block is included. IF condition fails with false. So second if which is outside of If is executed.

Test-5 Prog Logic with Solutions

1. What is the output of C Program.?

```
int main()
{
   if( 10 > 9 )
      printf("Singapore\n");
   else if(4%2 == 0)
      printf("England\n");
      printf("Poland");
   return 0;
}
```

A) Singapore
B) Singapore Poland
C) Singapore England Poland
D) England Poland

Answer [=] B
Explanation:
Observe that Poland printf is not under ELSE IF as there are two statements without curly braces { }. IF condition is TRUE. So, ELSE IF will not be seen at all even though 4%2 == 0 is true.

2. What is the output of C Program.?

```
int main()
{
   if("abc")
   {
      printf("India\n");
   }
   if('c')
   {
      printf("Honey\n");
   }
   printf("ZING");

   return 0;
}
```

A) ZING
B) Honey ZING
C) India ZING
D) India Honey ZING

Answer [=] D
Explanation:
"abc" is string and it returns an Integer Address Number. 'c' returns an ASCII number which is also a number. Any Non-Zero number gives TRUE condition.

3. What is the output of C Program.?
```
int main()
{
   if(TRUE)
   {
      printf("India\n");
   }
   if(true)
   {
      printf("Honey\n");
   }
   printf("ZING");

   return 0;
}
```

A) India ZING
B) Honey ZING
C) India Honey ZING
D) Compiler error

Answer [=] D
Explanation:
There are no keywords (true) or (TRUE). These available in Java, JavaScript and other languages.

4. What is the output of C Program.?
```
int main()
{
   int x=1;
```

```
    float y = 1.0;
    if(x == y)
    {
        printf("Polo\n");
    }
    if( 1 == 1.0)
    {
        printf("Golf\n");
    }

    if( 1.0 == 1.0f )
    {
         printf("Boxing\n");
    }
    return 0;
}
```

A) No Output
B) Boxing
C) Golf Boxing
D) Polo Golf Boxing

Answer [=] D
Explanation:
Integer is promoted to float or double automatically before comparison. So all are equal.
1 == 1.0 == 1.0f

5. What is the output of C Program.?

```
int main()
{
    int a=9;
    if(a=8)
    {
        printf("Kangaroo\n");
    }
    printf("Eggs\n");

    return 0;
}
```

A) No output
B) Eggs
C) Kangaroo Eggs
D) Compiler error

Answer [=] C
Explanation:
a=8 is an assignment not comparison. IF(Non Zero) is always TRUE.

6. What is the output of C Program.?
```
int main()
{
   int a=9;
   if(a==5);
   {
      printf("Kangaroo\n");
   }
   printf("Eggs\n");

   return 0;
}
```

A) Eggs
B) Kangaroo Eggs
C) No output
D) Compiler error

Answer [=] B
Explanation:
Notice a Semicolon at the end of IF(a==5);. So IF block ends without any statements. Also, { } is not part of IF now. So it is executed without checking for any condition.

7. What is the output of C Program.?
```
int main()
{
   int a=9, b=5, c=8;
   a=b=c=10;
   if(a==9)
   {
      printf("Ostrich\n");
```

```
    }
    else
    {
        printf("Eggs\n");
    }

    printf("White");

    return 0;
}
```

A) Ostrich Eggs White
B) Ostrich White
C) Eggs White
D) Compiler error as you can not assign to more than two variables at once.

Answer [=] C

8. What is the output of C Program.?

```
int main()
{
    int a=9, b=5, c=8;

    if(!(a==9))
    {
        printf("Bear\n");
    }
    else
    {
        printf("Elephant\n");
    }

    printf("Fox");

    return 0;
}
```

A) Bear Fox
B) Elephant Fox
C) Fox

D) Compiler error

Answer [=] B
Explanation:
Logical Not Operator (!) changes true to false and false to true. So IF(false) is not executed.

9. What is the Priority of C Logical Operators.? NOT (!), AND (&&) and OR (||)
A) NOT (!) > AND (&&) > OR (||)
B) NOT (!) > AND (&&) = OR (||)
C) AND (&&) > OR (||) > NOT (!)
D) AND (&&) = OR (||) > NOT (!)

Answer [=] A
Explanation:
Logical NOT Operator in C has the highest priority.

10. Which of the following are language processors?
A. Assembler
B. Compiler
C. Interpreter
D. All of the above

Answer: D

11. What is the output of C Program.?
```
int main()
{
   int a=9, b;

   b = (a==9) ? (printf("CAT\n");printf("DOG\n")) : (printf("FOX"));

   return 0;
}
```

A) CAT DOG
B) FOX
C) CAT DOG FOX
D) Compiler error

Answer [=] D
Explanation:
You can not put more than 1 statement inside expression1 or expression2 inside Ternary Operator.
(Condition)?(expression1):(expression2)

12. What is the output of C Program.?

```
int main()
{
   int a=9, b=6;
   if(a!=9 || b==6)
   {
      printf("Hockey\n");
   }
   else
   {
      printf("Cricket\n");
   }

   printf("Football");

   return 0;
}
```

A) Cricket Football
B) Hockey Football
C) Football
D) Compiler error

Answer [=] B
Explanation:
Logical OR || operator returns true if any one expression is true.

13. Choose a correct C Operator Priority.? Items in one group ( ) has same priority.

A) (!) < (*, /, %) < (+, -) < (<, <=, >, >=) < (==, !=) < (&&) < (||) < (=)
B) ((!) , (*, /, %) , (+, -)) < (<, <=, >, >=) < (==, !=) < (&&) < (||) < (=)

C) (!) > (*, /, %) > (+, -) > (<, <=, >, >=) > (==, !=) > (&&) > (||) > (=)
D) ((!) , (*, /, %) , (+, -)) > (<, <=, >, >=) > (==, !=) > (&&) > (||) > (=)

Answer [=] C
Explanation:
(! Logical NOT) > (*, /, % Arithmetic) > (+, - Arithmetic) > (<, <=, >, >= Relational) > (==, != Relational) > (&& Logical AND) > (|| Logical OR) > (= Assignment).

14. What is the output of C Program.?

```
int main()
{
   int a=5, b=8;

   if( a==5 && (b=9) )
   {
      printf("Gorilla Glass=");
   }
   printf("%d %d", a, b);

   return 0;
}
```

A) 5 8
B) 5 9
C) Gorilla Glass=5 8
D) Gorilla Glass=5 9

Answer [=] D
Explanation:
In IF(a==5 && (b=9)), && Operator checks both expressions for true or Non Zero value. So after checking a==5, b=9 is checked. Here b==9 is checking, but b=9 is assignment. Any NON-Zero value is true only. So b=9 now.

15. What is the output of C Program.?

```
int main()
{
   int a=5, b=8;

   if( a==5 || (b=9) )
   {
      printf("Gorilla Glass=");
```

```
    }
    printf("%d %d", a, b);

    return 0;
}
```

A) 5 8
B) 5 9
C) Gorilla Glass=5 8
D) Gorilla Glass=5 9

Answer [=] C
Explanation:
Logical OR || checks wants only one TRUE condition. First expression (a==5) or even (a=5) is true. So second expression (b=9) is not evaluated and assignment not done. So b=8 only.

Test-6 Prog Logic with Solutions

1. Choose a right C Statement.

A) Loops or Repetition block executes a group of statements repeatedly.
B) Loop is usually executed as long as a condition is met.
C) Loops usually take advantage of Loop Counter
D) All the above.

Answer [=] D

2. Which loop is faster in C Language, for, while or Do While.?
A) for
B) while
C) do while
D) All work at same speed

Answer [=] D

3. Choose a correct C for loop syntax.
A)
```
for(initalization; condition; incrementoperation)
{
   //statements
}
```
B)
```
for(declaration; condition; incrementoperation)
{
   //statements
}
```
C)
```
for(declaration; incrementoperation; condition)
{
   //statements
}
```
D)
```
for(initalization; incrementoperation; condition;)
{
   //statements
}
```

Answer [=] A
Explanation:
increment or decrement operation at third place.

4. Following while loop runs infinitely & causes your system to hang. What statement should you add after printf statement within while block to stop the while loop?

```
int main()
{
   int a=5;

   while(a==5)
   {
      printf("RABBIT");
      ________ ;
   }

   return 0;
}
```

A) continue;
B) break;
C) pause;
D) exit;

Answer [=] B
Explanation:
If there is no BREAK statement, while loop runs continuously until the computer hangs. BREAK causes the loop to break once and the statement below the while if any will be executed.

5. What is the output of C Program.? Hint: while(a=123) = while(123) = while(Non Zero Number). All of these are true.

```
int main()
{
   int a=5;

   while(a=123)
   {
      printf("RABBIT\n");
```

```
        break;
    }
    printf("GREEN");

    return 0;
}
```

A) GREEN
B) RABBIT GREEN
C) RABBIT is printed unlimited number of times.
D) Compiler error.

Answer [=] B
Explanation:
while(a=123) = while(123) = while(Non Zero Number). So while is executed. BREAK breaks the loop immediately. Without break statement, while loop runs infinite number of times.
6. What is the output of C Program.?

```
int main()
{
    int a=32;

    do
    {
        printf("%d ", a);
        a++;
    }while(a <= 30);

    return 0;
}
```

A) 32
B) 33
C) 30
D) No Output

Answer [=] A
Explanation:
do { } block is executed even before checking while(condition) at least once. This prints 32. To loop for the second time, while (32 <= 30) fails. So, loop is quit.

7. What is the output of C Program.?

```
int main()
{
   int a=32;

   do
   {
      printf("%d ", a);
      a++;
      if(a > 35)
         break;
   }while(1);

   return 0;
}
```

A) No Output
B) 32 33 34
C) 32 33 34 35
D) Compiler error

Answer [=] C
Explanation:
while(1) is infinite loop. So we kept if(condition) to break the loop. a++ is equivalent to a=a+1;

8. What is the output of C Program.?

```
int main()
{
   int k, j;

   for(k=1, j=10; k <= 5; k++)
   {
      printf("%d ", (k+j));
   }

   return 0;
}
```

A) compiler error

B) 10 10 10 10 10
C) 11 12 13 14 15
D) None of the above

Answer [=] C

Explanation:
You can initialize any number of variables inside for loop.

9. What is the output of C Program.? Hint: Notice the semicolon at the end of ‘for’ statement.

```
int main()
{
   int k;

   for(k=1; k <= 5; k++);
   {
      printf("%d ", k);
   }

   return 0;
}
```

A) 1 2 3 4 5
B) 1 2 3 4
C) 6
D) 5

Answer [=] C
Explanation:
Semicolon at the end of for(); isolates the below print() block. After for loop is over, k value is 6.

```
for(k=1; k <= 5; k++)
{
   ;
}
{
   printf("%d ", k);
}
```

10. What is the output of C Program.? Hint: for(;;) loop need not contain any initialization, condition and incre/decrement sections. All are optional. BREAK breaks the FOR Loop.

```
int main()
{
   int k;

   for(;;)
   {
      printf("TESTING\n");
      break;
   }

   return 0;
}
```

A) No Output
B) TESTING
C) Compiler error
D) None of the above

Answer [=] B
Explanation:
for(;;) loop need not contain any initialization, condition and incre/decrement sections. All are optional. BREAK breaks the FOR Loop.

11. What is the output of C Program.?
Tip: for(anything; anything; anything) is Ok. printf("YELLOW") prints YELLOW and returns 1 as result. So for loop runs forever. Actually break is saving us from quitting the for loop. Only after checking condition and executing the loop statements, third section is executed. break causes the loop to quit without incre/decrement section.

```
int main()
{
   int k;

   for(printf("FLOWER "); printf("YELLOW "); printf("FRUITS "))
   {
      break;
```

```
    }

    return 0;
}
```

A) Compiler error
B) FLOWER FRUITS
C) FLOWER YELLOW
D) FLOWER YELLOW FRUITS

Answer [=] C
Explanation:
for(anything; anything; anything) is Ok. printf("YELLOW") prints YELLOW and returns 1 as result. So for loop runs forever. Actually break is saving us quitting the for loop. Only after checking condition and executing the loop statements, third section is executed. break causes the loop to quit without incre/decrement section.

12. Structured programming languages are also called

A. Modular
B. Case sensitive
C. Pseudocode
D. Object Oriented language

Answer: A

13. In a for loop, if the condition is missing, then?

A. it is assumed to be present & taken to be false
B. it is assumed to be present & taken to be true
C. it results in a syntax error
D. execution will be terminated abruptly

Answer: D

14. A static variable by default gets initialized to

A. 0
B. blank space
C. 1

D. garbage value

Answer: D

15. Which of the following statement is correct about the array?

a. In the array, users can only allocate the memory at the run time.
b. In the array, users can only allocate the memory at the compile time.
c. The array is a primitive and non-linear data structure that only stores a similar data type.
d. All of the these

Answer: (b) In the array, users can only allocate the memory at the compile time.

Explanation: An array is a non-primitive and linear data structure that only stores a similar data type. In array, users can only allocate the memory at the compile time.

16. What will be the output of this program? Hint: '5' is equal to 53 in the ASCII table.

```
main ()
{
 if(5 < '5')
      printf("5")
 else
      printf("Not equal to 5.")
}
```

a. ENQ
b. 5
c. I
d. Not equal to 5

Answer: (b) 5
Explanation: This program will print 5 because '5' is a decimal value, and it is equal to 53 in the ASCII table. Therefore, the condition is true and returns 5.

17. Which of the following is the variable that can be used for all functions?

a. Static variable
b. Global variable
c. Local variable
d. Dynamic variable

Answer: (b) Global variable

Explanation: The global variable is a variable, which is declared outside of the functions. A global variable can be used in all functions.

18. What are included in the function prototype?
A. Function name, return type, data type of parameter
B. Function name, return type, parameters
C. Function call, received variable
D. Function return type, parameters & arguments

Answer: A,

Test-7 Prog Logic with Solutions

1. Which of the following initialization is incorrect in C language? Hint: Value to a string variable can be assigned in many ways.

a. char str [40] = "YUGAL";
b. char str [] = {'Y','U','G','A','L','\ 0'};
c. char str [40] = {'Y','U','G','A','L','\ 0'};
d. None of the these

Answer: (d) None of the these

Explanation: All these declarations are correct in the C language. Therefore, option (d) is the correct answer.

2. In C, parameters are always

A Passed by value
B Passed by reference
C Non-pointer variables are passed by value and pointers are passed by reference
D Passed by value result

Ans: A
Explanation: In C, function parameters are always passed by value.
Pass-by-reference is simulated in C by explicitly passing pointer values.

3. Which of the following is true about return type of functions in C?

A Functions can return any type
B Functions can return any type except array and functions
C Functions can return any type except array, functions and union
D Functions can return any type except array, functions, function pointer and union

Ans: B
In C, functions can return any type except arrays and functions.
We can get around this limitation by returning pointer to array or pointer to function.

4. What will be the output of the following code?

```
#include <stdio.h>
int main()
{
  printf("%d", main);
  return 0;
}
```

A Address of main function
B Compiler Error
C Runtime Error
D Some random value

Ans: A
Explanation: Name of the function is actually a pointer variable to the function and prints the address of the function.

5. Identify the result of this code:

```
#include<stdio.h>

void dynamic(int s, ...)
{
    printf("%d ", s);
}

int main()
{
    dynamic(2, 4, 6, 8);
    dynamic(3, 6, 9);
    return 0;
}
```

A 2 3
B Compiler Error
C 4 3
D 3 2

Ans: A
Explanation: In c three continuous dots is known as ellipsis which is variable number of

arguments of function. The values to parameters are assigned one by one.

Now the question is how to access other arguments?
C supports variable numbers of arguments. But there is no language provided way for finding out total number of arguments passed.

User has to handle this in one of the following ways:
1) By passing first argument as count of arguments.
2) By passing last argument as NULL (or 0).
3) Using some printf (or scanf) like mechanism where first argument has placeholders for rest of the arguments.

6. What is the meaning of using static before function declaration?

```
static int sum(int x, int y, int z)
{
   return (x + y + z);
}
```

A Static means nothing, sum() is same without static keyword.
B Function need not to be declared before its use
C Access to static functions is restricted to the file where they are declared
D Static functions are made inline

Ans: C
Explanation: In C, functions are global by default. Unlike global functions, access to static functions is restricted to the file where they are declared.
We can have file level encapsulation using static variables/functions in C because when we make a global variable static, access to the variable becomes limited to the file in which it is declared.

7. In C, what is the meaning of following function prototype with empty parameter list

```
void fun()
{
  /* .... */
}
```

A Function can only be called without any parameter
B Function can be called with any number of parameters of any types

C Function can be called with any number of integer parameters.
D Function can be called with one integer parameter.

Ans: B
Explanation: Empty list in C mean that the parameter list is not specified and function can be called with any number of parameters. In C, to declare a function that can only be called without any parameter,   we should use "void fun(void)"
As a side note, in C++, empty list means function can only be called without any parameter.
In C++, both void fun() and void fun(void) are same.

8. What is required in each C program?

A. The program must have at least one function
B. The program does not require any function
C. Input data
D. Output data

Ans: A
Explanation: Any C program has at least one function, and even the most trivial programs can specify additional functions.
A function is a piece of code. In other words, it works like a sub-program.

9. Which one of the following is a loop construct that will always be executed once?

A. for
B. while
C. switch
D. do while

Ans: D
Explanation: The body of a loop is often executed at least once during the do-while loop. Once the body is performed, the condition is tested.
If the condition is valid, it will execute the body of a loop;
otherwise, control is transferred out of the loop.

10. Find correct output?

```
#include <stdio.h>
int Calculate(int x, int y) {
        int y=10;
        return x + y;
}

int main() {
  int result = Calculate(15, 3);
  printf("Result is = %d", result);

  return 0;
}
```

A 18
B 25
C 13
D Compilation error

Answer: D
Explanation: error: ‘y’ redeclared as different kind of symbol
note: previous definition of ‘y’ was here

```
     int Calculate(int x, int y)
```

11. What is x in the following program?

```
#include<stdio.h>
int main ()
{
typedef char (*(*arrfptr[3])())[10];
arrfptr x  return 0;  }
```

A. x is a character pointer
B. x is an array of pointer
C. x is an array of three function pointers
D. Wrong declaration

Ans: C
Explanation: Function Pointers point to code like normal pointers.
In Functions Pointers, the function’s name can be used to get the function’s address.
Here, x is an array of three function pointers i.e. typedef char (*(*arrfptr[3])())[10];

12. What is the output of the following program?

```
#include<stdio.h>
main()
{
int a[3] = {2,1};
printf("%d", a[a[1]]);
}
```

A. 0
B. 1
C. 2
D. 3

Ans: B
Explanation: The given program is error-free and execute without throwing any error.
1, The inner indirection i.e. a[i] evaluates to 1, and the value at index 1 for outer indirection is 1.
Thus the output will be 1.

13. Study the following array definition
int num[10] = {3, 3, 3};
Which of the following statement is correct?

A. num[9] is the last element of the array num
B. The value of num[8] is 3
C. The value of num[3] is 3
D. None of the above

Ans: A
Explanation: The num[9] is the last element of the array number because the total element in this array is 10, and the array starts with 0, so the last element of the array is the num[9].

14. Better practice: How to optimize this code?

```
#include <stdio.h>
int Calculate(int x, int y) {
        return x + y;
}

int main() {
  int result = Calculate(15, 3);
  printf("Result is = %d", result);

  return 0;
}
```

A. Declare function before main() & Define function after main()
B. Define function before main() & Declare function after main()
C. No optimization is required
D. Move Function definition to after main()

Ans: A

15. Which of the following is the correct format for declaration of function?

A. return-type function-name(argument type);
B. return-type function-name(argument type){}
C. return-type (argument type)function-name;
D. all of the mentioned

Answer: A

Question	**1. Study the following array definition int num[10] = {3, 3, 3}; Which of the following statement is correct?** A. num[9] is the last element of the array num B. The value of num[8] is 3 C. The value of num[3] is 3 D. None of the above	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	**Explanation:** The num[9] is the last element of the array number because the total element in this array is 10, and the array starts with 0, so the last element of the array is the num[9].	
Marks	1	

Question	**2. Better practice: How to optimize this code?**
	
	#include <stdio.h>
	int Calculate(int x, int y) {
	        return x + y;
	}
	
	int main() {
	  int result = Calculate(15, 3);
	  printf("Result is = %d", result);
	
	  return 0;
	}
	
	A. Declare function before main() & Define function after main()
	B. Define function before main() & Declare function after main()
	C. No optimization is required

	D. Move Function definition to after main()	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	**3. Which of the following is the correct format for declaration of function?**
	
	A. return-type function-name(argument type);
	B. return-type function-name(argument type){}
	C. return-type (argument type)function-name;
	D. all of the mentioned	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	**4. Can we use a function as a parameter of another function? [Eg: void wow(int func())].** a) Yes, and we can use the function value conveniently b) Yes, but we call the function again to get the value, not as convenient as in using variable c) No, C does not support it

	d) This case is compiler dependent	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	**5. What will be the output of the following C code?**
	`#include <stdio.h>`
	`#include <math.h>`
	`void main()`
	`{`
	`    int k = sqrt(-4);`
	`    printf("%d\n", k);`
	`}`
	
	a) -2
	b) 2
	c) Compile time error
	d) NaN	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect

Solution	Not a Number	
Marks	1	

Question	**6. Which function definition will run correctly?** a) int sum(int a, int b) return (a + b); b) int sum(int a, int b) {return (a + b);} c) int sum(a, b) return (a + b); d) none of the mentioned	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	7. **Add a name parameter of type char (string) to myFunction.**
	
	void greetName(____ ______)
	{
	  printf("Hello %s\n", name);
	}
	
	int main() {
	  greetName("Liam");
	  greetName("Jenny");
	  greetName("Anja");
	  return 0;
	}

	A. char[] name B. string name[] C. char name[] D. char name	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	8. **What will be the Output of the following C Code?**
	`#include <stdio.h>`
	`int Calculate(int a, int b) {`
	`        int y = 7;`
	`        return a + b;`
	`}`
	
	`int main() {`
	`  int result = Calculate(10, 5);`
	`  printf("Result is = %d", result);`
	
	`  return 0;`
	`}`
	
	A 15
	B 17
	C 22
	D Compilation error	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect

Solution	Explanation: error: ‘b’ redeclared as different kind of symbol note: previous definition of ‘y’ was here int Calculate(int a, int b)	
Marks	1	

Question	**9. Which of the following statement is correct?**
	
	double a, b, c;
	a = 17.797979;
	c = modf(a, &b);
	
	a) b stores integer part of a, c returns integer part of a
	
	b) b stores integer part of a, c returns fractional part of a
	
	c) b stores fractional part of a, c returns integer part of a
	
	d) b stores fractional part of a, c returns fractional part of a	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	**Explanation:** double modf(double x, double *y) This function returns the fractional part of x, with the same sign. modf() function breaks the argument value into integer and fraction parts, each of which has the same sign as the argument. It stores the integer part as a double in the object pointed to by y.	
Marks	1	

Question	10. **Which of the following is the variable that can be used for all functions?** a. Static variable b. Global variable c. Local variable d. Dynamic variable	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	**Global variable**	
Marks	1	

Question	11. **Determine output:**
	`main()`
	`{`
	`        int i = abc(10);`
	`        printf("%d", --i);}`
	`int abc(int i)`
	`{`
	`        return(i++);}`

	A.10 B.9 C.11 D.None of these.	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	Solution: return(i++) it will first return i and then increment. i.e. 10 will be returned.	
Marks	1	

Question	**12. What will this proggram snippet print?**
	main()
	{
	        char string[] = "Hello World";
	        display(string);}
	
	void display(char *string)
	{
	        printf("%s", string);}
	
	A.will print Hello World
	B.Compilation Error
	C.will print garbage value

	D.None of these.	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	Solution: Compiler Error : Type mismatch in redeclaration of function display As the function display() is not defined before use, compiler will assume the return type of the function which is int(default return type). But when compiler will see the actual definition of display(), mismatch occurs since the function display() is declared as void. Hence the error.	
Marks	1	

Question	13. **What will be printed when this program is executed?**
	`int f(int x)`
	`{`
	`  if(x <= 5)`
	`          return x;`
	`  return f(--x);}`
	`void main()`
	`{`
	`  printf("%d ", f(9)); }`
	
	A. 5 6 7 8 9
	B. 1 2 3 4 5
	C. 5
	D. Syntax error	

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect

Solution	Solution: In this recursive function call the function will return to main caller when the value of x is 4. Hence the output.	
Marks	1	

Question	**14. The recursive functions are executed in .... order.** A.Parallel order B.First In First Out order C.Last In First Out order D.Iterative order	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	Solution: Because for each function call an entry is created in stack frame(known as Active Record Instance), and are executed in LIFO manner.	
Marks	1	

Question	15. **What will be printed when this sample code is executed?**
	
	` char* myfunc(char *ptr){`
	`   ptr+=3;`
	`   return(ptr);}`
	`void main()`
	`{`
	`   char *x, *y;`
	`   x = "EXAMVEDA";`
	`   y = myfunc(x);`
	`   printf("y=%s", y);}`
	
	A. y=EXAMVEDA
	B. y=MVEDA
	C. y=VEDA
	D. y=EDA

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	Solution: ptr+3 increments pointer location by 3 char units	
Marks	1	

Test-8 Prog Logic With Solutions

1. Can we use a function as a parameter of another function? [Eg: void wow(int func())].

a) Yes, and we can use the function value conveniently

b) Yes, but we call the function again to get the value, not as convenient as in using variable

c) No, C does not support it

d) This case is compiler dependent

Answer: c

2. What will be the output of the following C code?

```
#include <stdio.h>

#include <math.h>

void main()

{

    int k = sqrt(-4);

    printf("%d\n", k);

}
```

a) -2

b) 2

c) Compile time error

d) NaN

Answer: d (Not a number)

3. Which function definition will run correctly?

a)
```
int sum(int a, int b)
return (a + b);
```
b)
```
int sum(int a, int b)
{return (a + b);}
```
c)
```
int sum(a, b)
return (a + b);
```

d) none of the mentioned

Answer: b

4. Add a name parameter of type char (string) to myFunction.

```
void greetName(____ ______)
{
  printf("Hello %s\n", name);
}

int main() {
  greetName("Liam");
  greetName("Jenny");
  greetName("Anja");
  return 0;
}
```

A. char[] name
B. string name[]
C. char name[]
D. char name

Answer: C

5. What will be the Output of the following C Code?

```
#include <stdio.h>
int Calculate(int a, int b) {
        int y = 7;
        return a + b;
}

int main() {
  int result = Calculate(10, 5);
  printf("Result is = %d", result);

  return 0;
}
```

A 15
B 17
C 22
D Compilation error

Answer: D
error: ‘b’ redeclared as different kind of symbol
note: previous definition of ‘y’ was here
    int Calculate(int a, int b)

6. Which of the following statement is correct?

```
double a, b, c;

 a = 17.797979;

 c = modf(a, &b);
```

a) b stores integer part of a, c returns integer part of a

b) b stores integer part of a, c returns fractional part of a

c) b stores fractional part of a, c returns integer part of a

d) b stores fractional part of a, c returns fractional part of a

Answer: b

Explanation: double modf(double x, double *y)

This function returns the fractional part of x, with the same sign. modf() function breaks the argument value into integer and fraction parts, each of which has the same sign as the argument. It stores the integer part as a double in the object pointed to by y.

7. Which of the following is the variable that can be used for all functions?

a. Static variable
b. Global variable
c. Local variable
d. Dynamic variable

Answer: (b) Global variable

Explanation: The global variable is a variable, which is declared outside of the functions. A global variable can be used in all functions.

8. What syntax do you use to get the value of a pointer *ptr?

A. *ptr
B. ptr
C. &ptr
D. All of the above

Answer: A

Explanation: *ptr returns the value; ptr returns the address it points to; &ptr returns self-address

9.

Test-9 CRT Technical Practice

Question	**1. Which of the following operator takes only integer operands?** A. + B. * C. / D. %	
Type	multiple_choice	
Option	D	correct
Option	B	incorrect
Option	C	incorrect
Option	A	incorrect
Solution	Two integers are taken to be input	
Marks	1	

Question	**2.In an expression involving || operator, evaluation** **I. Will be stopped if one of its components evaluates to false** **II. Will be stopped if one of its components evaluates to true** **III. Takes place from right to left** **IV. Takes place from left to right** **A. II and IV** **B. I and III** **C. II and III** **D. I and II**	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect

Option		incorrect
Solution		
Marks	1	

Question	**3.Determine output:** **void main()** **{** **int i=0, j=1, k=2, m;** **m = i++ || j++ || k++;** **printf("%d %d %d %d", m, i, j, k);}** A. 1 1 2 2 B. 1 1 2 3 C. 0 1 2 2 D. 0 1 2 3	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	In an expression involving || operator, evaluation takes place from **left to right** and will be stopped if one of its components evaluates to true(a non zero value). So in the given expression m = i++ || j++ || k++. It will be stop at j and assign the current value of j in m. therefore m = 1 , i = 1, j = 2 and k = 2 **(since k++ will not encounter. so its value remain 2)**	
Marks	1	

Question	**4.Determine output:**
	void main()
	{
	int c = - -2;
	printf("c=%d", c); }

	A. 1 **B. -2** **C. 2** **D. Error**	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	Here unary minus (or negation) operator is used twice. Same maths rules applies, ie. minus * minus = plus. Note: However you cannot give like --2. Because -- operator can only be applied to variables as a decrement operator (eg., i--). 2 is a constant and not a variable.	
Marks	1	

Question	**5.Determine output:** **void main()** **{** **int i=10;** **i = !i>14;** **printf("i=%d", i); }** A. 10 B. 14 C. 0 D. 1	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect

Option	B	incorrect
Option	D	incorrect
Solution	In the expression !i>14 , **NOT (!)** operator has more precedence than ' **>**' symbol. **!** is a unary logical operator. **!i** (!10) is 0 (not of true is false). 0>14 is false (zero).	
Marks	1	

Question	**6.In C programming language, which of the following type of operators have the highest precedence** A. Arithmetic operators B. Equality operators C. Logical operators D. Relational operators	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	**7. What will be the output of the following program?**
	void main()
	{
	int a, b, c, d;
	    a = 3;
	    b = 5;
	    c = a, b;
	    d = (a, b);
	printf("c=%d d=%d", c, d);}
	
	A. c=3 d=3

	B. c=3 d=5 **C. c=5 d=3** **D. c=5 d=5**	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	The comma operator evaluates both of its operands and produces the value of the second. It also has lower precedence than assignment. Hence c = a, b is equivalent to c = a, while d = (a, b) is equivalent to d = b.	
Marks	1	

Question	**8. Which of the following comments about the ++ operator are correct?** A. It is a unary operator B. The operand can come before or after the operator C. It cannot be applied to an expression & It associates from the right D. All of the above	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	**9. What will be the output of this program on an implementation where int occupies 2 bytes?**
	#include <stdio.h>void main()
	{
	int i = 3;
	int j;

	j = sizeof(++i + ++i); **printf("i=%d j=%d", i, j);}** **A. i=4 j=2** **B. i=3 j=2** **C. i=5 j=2** **D. the behavior is undefined**	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	Evaluating ++i + ++i would produce undefined behavior, but the operand of sizeof is not evaluated, so i remains 3 throughout the program. The type of the expression (int) is reduced at compile time, and the size of this type (2) is assigned to j.	
Marks	1	

Question	**10.Which operator has the lowest priority?** A. ++ B. % C. + D. ||	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	See C Operator Precedence Table.	

Marks	1	

Question	**11. What will be the output?**
	void main(){
	int a=10, b=20;
	char x=1, y=0;
	if(a,b,x,y){
	printf("EXAM");
	
	A. XAM is printed
	
	B. exam is printed
	
	C. Compiler Error
	
	D. Nothing is printed	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	**12. What number will z in the sample code given below?**
	int z, x=5, y= -10, a=4, b=2;
	z = x++ - --y*b/a;

	A. 5 B. 6 C. 9 D. 10	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	C Operator Precedence Table According to precedence table execution of the given operators are as follows: 1. x++(Postfix operator) i.e x will become 5 2. y--(Prefix operator) i.e y will become -11 3. * and / have same priority so they will be executed according to their associativity i.e left to right. So, *(Multiplication) will execute first and then /(division). 4. -(Subtraction) So the complete expression would be 5 - (-11)*2/4 = 5 - (-22)/4 = 5 - (-5) = 5 + 5 = 10.	
Marks	1	

Question	**13. What is the output of the following statements?** **int i = 0;printf("%d %d", i, i++);** A. 0 1 B. 1 0 C. 0 0 D. 1 1
Type	multiple_choice

Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	Since the evaluation is from **right to left.** So when the print statement execute value of *i* = 0 Since its execute from right to left when *i++* will be execute first and print value 0 **(since its post increment)** and after printing 0 value of *i* become 1. **So it its prints for 1 for next *i*.**	
Marks	1	

Question	**14. What is the output of the following statements?** **int b=15, c=5, d=8, e=8, a;** **a = b>c ? c>d ? 12 : d>e ? 13 : 14 : 15;printf("%d", a);** **A. 13** **B. 14** **C. 15** **D. 12**	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect

Solution	Expression
	a = b>c ? c>d ? 12 : d>e ? 13 : 14 : 15;
	can be rewritten as
	if(b>c)
	{
	if(c>d)
	    a = 12;
	else
	{
	if(d>e)	

	a = 13;
	else
	a = 14;
	}
	}else{
	a = 15;}	

Marks	1	

Question	**15. What will be the output of the following code fragment?** **void main()** **{** **printf("%x",-1<<4);}** **A. fff0** **B. fff1** **C. fff2** **D. fff3**	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	-1 will be represented in binary form as: 1111 1111 1111 1111 Now -1<<4 means 1 is Shifted towards left by 4 positions, hence it becomes: 1111 1111 1111 0000 in hexadecimal form - fff0.	
Marks	1	

Question	**16. Find the output of the following program.**
	#includevoid main()
	{
	int y=10;
	if(y++>9 && y++!=10 && y++>11)
	printf("%d", y);	

	else **printf("%d", y);** **}** **A. 11** **B. 12** **C. 13** **D. 14**	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	All the three condition in ***if*** is true. if(y++>9 && y++!=10 && y++>11) such as **1st condition :** 10++ > 9 is true and value of y is increase by 1 **i.e y =11** **2nd condition :** 11++ != 10 is also ture and value of y is increase by 1 **i.e y =12** **3rd condition :** 12++ > 11 is also ture and value of y is increase by 1 **i.e y =13** Therefore ***if*** is excuted and print the value of **y = 13**	
Marks	1	

Question	**17. Find the output of the following program.**
	#include<stdio.h>void main()
	{
	int y=10;
	if(y++>9 && y++!=11 && y++>11)
	printf("%d", y);
	else
	printf("%d", y);
	}
	
	A. 11
	
	B. 12

	C. 13 **D. 14**	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	Since the second condition is false so, further conditions will not be checked, it will be skipped.	
Marks	1	

Question	**18. Determine output of the following program code.**
	
	#include<stdio.h>void main()
	{
	int a, b=7;
	** a = b<4 ? b<<1 : ++b>4 ? 7>>1 : a;**
	printf("%d %d", a, b);}
	
	A. 3 7
	
	B. 7 3
	
	C. 8 3
	
	D. 3 8	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	**19.Choose the correct output for the following program.** **#include<stdio.h>void main()** **{** **int a=10, b=11, c=13, d;** **d = (a=c, b+=a, c=a+b+c);** **printf("%d %d %d %d", d, a, b, c);}** A.50, 13, 11, 13 B.50, 13, 24, 50 C.13, 10, 24, 50 D.50, 13, 24, 13	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	For any comma separated expression the outcome is the right most part	
Marks	1	

Question	**20. Consider the following program fragment, and choose the correct one
	void main()
	{
	int a, b = 2, c;
	  a = 2 * (b++);
	  c = 2 * (++b);}
	A.a = 4, c = 8
	B.a = 3, c = 8
	C.b = 3, c = 6
	D.a = 4, c = 6**	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	**21. Which operator from the following has the lowest priority? A.Assignment operator B.Division operator C.Comma operator D.Conditional operator**	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	**22. Identify the correct output of the following code:**

	void main() **{** **int w=10, x=5, y=3, z=3;** **if((w < x) && (y=z++))** **printf("%d %d %d %d", w, x, y, z);** **else** **printf("%d %d %d %d", w, x, y, z);}** **A.10 5 4 4** **B.10 5 3 3** **C.10 5 4 3** **D.10 5 3 4**	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	As the first condition (w < x) is false and the logical operator && is used, the entire relation becomes false and the second part (y = z++) is not executed.	
Marks	1	

Question	**23. Given b=110 and c=20, what is the value of 'a' after execution of the expression a=b-=c*=5?** **A.450** **B.10** **C.110** **D.-10**	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	As the expression is evaluated from right to left.	
Marks	1	

Question	**24. void main()
	{
	int a=10, b;
	   b = a++ + ++a;
	printf("%d %d %d %d", b, a++, a, ++a);}
	what will be the output when following code is executed?
	
	A.12 10 11 13
	B.22 12 12 13
	C.22 11 11 11
	D. .22 13 14 14**	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	**25. Determine Output:
	
	voidmain() {
	  int i = 0, j = 1, k = 2, m;
	  m = i++ || j++ || k++;
	  printf("%d %d %d %d", m, i, j, k);
	}
	a) 1 1 2 3
	b) 1 1 2 2
	c) 0 1 2 2
	d) 0 1 2 3
	e) None of these**	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect

Solution		
Marks	1	

Test-10 CRT C Pointers-A

Question	1. What will be the output of the following C code?
	  #include <stdio.h>
	  int main()
	  {
	      char *p = NULL;
	      char *q = 0;
	      if (p)
	          printf(" p ");
	      else
	          printf("nullp");
	      if (q)
	          printf("q\n");
	      else
	          printf(" nullq\n");
	  }
	
	a) nullp nullq
	b) Depends on the compiler
	c) x nullq where x can be p or nullp depending on the value of NULL
	d) p q	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	2. What will be the output of the following C code?
	
	  #include <stdio.h>
	  int main()
	  {
	      int i = 10;
	      void *p = &i;
	      printf("%d\n", (int)*p);
	      return 0;
	  }

	a) Compile time error b) Segmentation fault/runtime crash c) 10 d) Undefined behaviour	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	3. What will be the output of the following C code?
	
	`#include <stdio.h>`
	`int main()`
	`{`
	`    int i = 10;`
	`    void *p = &i;`
	`    printf("%f\n", *(float*)p);`
	`    return 0;`
	`}`
	
	a) Compile time error
	b) Undefined behaviour
	c) 10
	d) 0.000000	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	4. What will be the output of the following C code?
	
	`#include <stdio.h>`
	`int *f();`
	`int main()`
	`{`
	`    int *p = f();`
	`    printf("%d\n", *p);`
	`}`
	`int *f()`
	`{`
	`    int *j = (int*)malloc(sizeof(int));`
	`    *j = 10;`
	`    return j;`
	`}`
	
	a) 10
	b) Compile time error
	c) Segmentation fault/runtime crash since pointer to local variable is returned
	d) Undefined behaviour	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	5. What will be the output of the following C code?
	
	`#include <stdio.h>`
	`int *f();`
	`int main()`
	`{`
	`    int *p = f();`
	`    printf("%d\n", *p);`
	`}`

	int *f()
	{
	    int j = 10;
	    return &j;
	}
	
	a) 10
	b) Compile time error
	c) Segmentation fault/runtime crash
	d) Undefined behaviour	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	6. What will be the output of the following C code?
	  #include <stdio.h>
	  int main()
	  {
	      int *ptr, a = 10;
	      ptr = &a;
	      *ptr += 1;
	      printf("%d,%d/n", *ptr, a);
	  }
	a) 10,10
	b) 10,11
	c) 11,10
	d) 11,11	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect

Option	C	incorrect
Solution		
Marks	1	

Question	7. Which of the following does not initialize ptr to null (assuming variable declaration of a as int a=0;)?
	
	a) int *ptr = &a;
	
	b) int *ptr = &a – &a;
	
	c) int *ptr = a – a;
	
	d) All of the mentioned	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	8. What will be the output of the following C code?
	`#include <stdio.h>`
	`int x = 0;`
	`void main()`
	`{`
	`    int *ptr = &x;`
	`    printf("%p\n", ptr);`
	`    x++;`
	`    printf("%p\n ", ptr);`
	`}`
	a) Same address
	b) Different address

	c) Compile time error d) Varies	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	9. What will be the output of the following C code?
	
	`#include <stdio.h>`
	`void main()`
	`{`
	`    int x = 0;`
	`    int *ptr = &x;`
	`    printf("%p\n", ptr);`
	`    ptr++;`
	`    printf("%p\n ", ptr);`
	`}`
	
	a) 0xbfd605e8 0xbfd605ec
	b) 0xbfd605e8 0cbfd60520
	c) 0xbfd605e8 0xbfd605e9
	d) Run time error	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	10. What will be the output of the following C code?
	`#include <stdio.h>`
	`void foo(int*);`
	`int main()`
	`{`
	`    int i = 10;`
	`    foo((&i)++);`
	`}`
	`void foo(int *p)`
	`{`
	`    printf("%d\n", *p);`
	`}`
	
	a) 10
	b) Some garbage value
	c) Compile time error
	d) Segmentation fault/code crash	

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	11. What will be the output of the following C code?
	`#include <stdio.h>`
	`int main()`
	`{`
	`    int i = 97, *p = &i;`
	`    foo(&i);`
	`    printf("%d ", *p);`
	`}`
	`void foo(int *p)`
	`{`

```
      int j = 2;
      p = &j;
      printf("%d ", *p);
  }
```

a) 2 97
b) 2 2
c) Compile time error
d) Segmentation fault/code crash

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	12. What will be the output of the following C code?

```
  #include <stdio.h>
  int main()
  {
      int i = 97, *p = &i;
      foo(&p);
      printf("%d ", *p);
      return 0;
  }
  void foo(int **p)
  {
      int j = 2;
      *p = &j;
      printf("%d ", **p);
  }
```

a) 2 2
b) 2 97
c) Undefined behaviour
d) Segmentation fault/code crash

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	13. What will be the output of the following C code?
	
	`#include <stdio.h>`
	`int main()`
	`{`
	`    int i = 10;`
	`    int *p = &i;`
	`    foo(&p);`
	`    printf("%d ", *p);`
	`    printf("%d ", *p);`
	`}`
	`void foo(int **const p)`
	`{`
	`    int j = 11;`
	`    *p = &j;`
	`    printf("%d ", **p);`
	`}`
	
	a) 11 11 11
	b) 11 11 Undefined-value
	c) Compile time error
	d) Segmentation fault/code-crash	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect

Option	D	incorrect
Solution		
Marks	1	

Question	14. Which of the following can never be sent by call-by-value? a) Variable b) Array c) Structures d) Both Array and Structures	
Type	multiple_choice	
Option	B	correct
Option	C	incorrect
Option	D	incorrect
Option	A	incorrect
Solution		
Marks	1	

Question	15. Which type of variables can have the same name in a different function? a) Global variables b) Static variables c) Function arguments d) Both static variables and Function arguments	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	16. What will be the output of the following C code?
	`#include <stdio.h>`
	`void m(int *p, int *q)`
	`{`
	`    int temp = *p; *p = *q; *q = temp;`
	`}`
	`void main()`
	`{`
	`    int a = 6, b = 5;`
	`    m(&a, &b);`
	`    printf("%d %d\n", a, b);`
	`}`
	
	a) 5 6
	b) 6 5
	c) 5 5
	d) 6 6	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	17. What will be the output of the following C code?
	`#include <stdio.h>`
	`void m(int *p)`
	`{`
	`    int i = 0;`
	`    for(i = 0;i < 5; i++)`
	`    printf("%d\t", p[i]);`
	`}`
	`void main()`
	`{`

	```
	    int a[5] = {6, 5, 3};
	    m(&a);
	  }
	```
	a) 0 0 0 0 0
	b) 6 5 3 0 0
	c) Run time error
	d) 6 5 3 junk junk	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	18. What will be the output of the following C code?
	```
	  #include <stdio.h>

	  void main()
	  {
	    int a[3] = {1, 2, 3};
	    int *p = a;
	    printf("%p\t%p", p, a);
	  }
	```
	a) Same address is printed
	
	b) Different address is printed
	
	c) Compile time error
	
	d) Nothing

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	19. What will be the output of the following C code?
	`#include <stdio.h>`
	`void main()`
	`{`
	`    char *s= "hello";`
	`    char *p = s;`
	`    printf("%c\t%c", p[0], s[1]);`
	`}`
	
	a) Run time error
	b) h h
	c) h e
	d) h l	

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	20. What will be the output of the following C code?

	```
	#include <stdio.h>
	void main()
	{
	    char *s= "hello";
	    char *p = s;
	    printf("%c\t%c", *(p + 3),  s[1]);
	}
	```
	 a) h e
	b) l l
	c) l o
	d) l e	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	1. What will be the output of the following C code?
	`#include <stdio.h>`
	`void main()`
	`{`
	`    char *s= "hello";`
	`    char *p = s;`
	`    printf("%c\t%c", 1[p], s[1]);`
	`}`
	
	a) h h
	b) Run time error
	c) l l
	d) e e	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	Pointer array can be written as p[i] or i[p]	
Marks	1	

Question	2. What will be the output of the following C code?
	
	`#include <stdio.h>`
	`void foo( int[] );`
	`int main()`
	`{`
	`    int ary[4] = {1, 2, 3, 4};`
	`    foo(ary);`
	`    printf("%d ", ary[0]);`
	`}`
	`void foo(int p[4])`
	`{`
	`    int i = 10;`
	`    p = &i;`
	`    printf("%d ", p[0]);`
	`}`

	a) 10 10 b) Compile time error c) 10 1 d) Undefined behaviour	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	3. What will be the output of the following C code?
	`#include <stdio.h>`
	`int main()`
	`{`
	`    int ary[4] = {1, 2, 3, 4};`
	`    int *p = ary + 3;`
	`    printf("%d\n", p[-2]);`
	`}`
	
	a) 1
	b) 2
	c) Compile time error
	d) Some garbage value	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect

Solution		
Marks	1	

Question	4. What will be the output of the following C code?
	`#include <stdio.h>`
	`int main()`
	`{`
	`    int ary[4] = {1, 2, 3, 4};`
	`    int *p = ary + 3;`
	`    printf("%d %d\n", p[-2], ary[*p]);`
	`}`
	
	a) 2 3
	b) Compile time error
	c) 2 4
	d) 2 somegarbagevalue	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	5. What will be the output of the following C code?
	`#include <stdio.h>`
	`int main()`
	`{`
	`    int ary[4] = {1, 2, 3, 4};`
	`    print("%d\n", *ary);`
	`}`
	
	a) 1
	b) Compile time error
	c) Some garbage value
	d) Undefined variable

Type	multiple_choice

Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		

<table>
<tr><td>Question</td><td colspan="2">6. Find the output?<pre>
 #include <stdio.h>
   int main()
   {
       const int ary[4] = {1, 2, 3, 4};
       int *p;
       p = ary + 3;
       *p = 5;
       printf("%d\n", ary[3]);
   }
</pre>
a) 4
b) 5
c) Compile time error
d) 3</td></tr>
<tr><td>Type</td><td colspan="2">multiple_choice</td></tr>
<tr><td>Option</td><td>B</td><td>correct</td></tr>
<tr><td>Option</td><td>A</td><td>incorrect</td></tr>
<tr><td>Option</td><td>C</td><td>incorrect</td></tr>
<tr><td>Option</td><td>D</td><td>incorrect</td></tr>
<tr><td>Solution</td><td colspan="2"></td></tr>
<tr><td>Marks</td><td>1</td><td></td></tr>
</table>

<table>
<tr><td>Question</td><td>7. <b>What are the different ways to initialize an array with all elements as zero?</b>

A. int array[5] = {0};

B. int a = 0, b = 0, c = 0;   int array[5] = {a, b, c};

C. int array[5] = {};</td></tr>
</table>

	D. All of the above	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	8. **What will be the output of the following C code?**
	
	void main()
	{
	int a[] = {1,2,3,4,5}, *p;
	p = a; ++*p; **// Incremented VALUE by 1**
	printf("%d ", *p);
	p += 2; **// Incremented ADDRESS by 2 units**
	printf("%d ", *p);
	}
	
	A. 2 2
	B. **2 3**
	C. 2 4
	D. 3 4	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		

Marks	1	

Question	9. **Identify the output?**
	```int *f();```
	```int main()```
	```{```
	```int *p = f();```
	```printf("%d\n", *p);```
	```}```
	
	```int *f()```
	```{```
	```int *j = (int*)malloc(sizeof(int));```
	```*j = 10;```
	```return j;```
	```}```
	
	A. 10
	
	B. Compile time error
	
	C. Segmentation fault/runtime crash since pointer to local variable is returned
	
	D. Undefined behaviour	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	10. **What will be the output of the following C code?** **void main()**

	{
	int a[] = {1,2,3,4,5}, *p;
	p = a; ++*p;
	printf("%d ", *p);
	p += 2;
	printf("%d ", *p);
	}
	
	A. 2 2
	B. 2 3
	C. 2 4
	D. 3 4	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	11. **Comment on the following?** **const int *ptr;** A. You cannot change the value pointed by ptr B. You cannot change the pointer ptr itself C. Both (a) and (b) D. You can change the pointer as well as the value pointed by it

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	12. **Identify the output?**
	#include<stdio.h>
	int main()
	{
	static char *s[] = {"black", "white", "yellow", "violet"};
	char **ptr[] = {s+3, s+2, s+1, s}, ***p;
	p = ptr;
	++p; **//points to s+2 i.e., pink**
	printf("%s", **p+2); **//points to i, the next char in pink**
	return 0;
	}
	A. let
	B. ite
	C. llow
	D. ack	

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	13. **Identify the output?** **/* Assume address of x is 500 and integer is 4 byte size */**
	`#include<stdio.h>`
	`int main()`
	`{`
	`int x=30, *y, *z;`
	`y=&x;`
	`z=y;`
	`*y++=*z++;`
	`x++;`
	`printf("x=%d, y=%d, z=%d\n", x, y, z);`
	`return 0;`
	`}`
	
	A. x=31, y=504, z=504
	B. x=31, y=498, z=498
	C. x=31, y=500, z=500
	D. x=31, y=502, z=502	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	14. What are the elements present in the array of the following C code? `int array[5] = {5};` a) 5, 5, 5, 5, 5

	b) 5, 0, 0, 0, 0 c) 5, (garbage), (garbage), (garbage), (garbage)	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	15. An array of similar data types which themselves are a collection of dissimilar data type are ____________ a) Linked Lists b) Trees c) Array of Structure d) All of the mentioned	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution		

Marks	1	

Question	16. What will be the output of the following C code?
	
	`#include <stdio.h>`
	`int main()`
	`{`
	`    double *ptr = (double *)100;`
	`    ptr = ptr + 2;`
	`    printf("%u", ptr);`
	`}`
	
	a) 102
	
	b) 104
	
	c) 108
	
	d) 116	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	17. What will be the output of the following C code?
	
	`#include <stdio.h>`
	`int main()`
	`{`
	`    int *p = (int *)2;`
	`    int *q = (int *)3;`
	`    printf("%d", p + q);`
	`}`
	
	a) 2
	
	b) 3
	
	c) 5
	
	d) Compile time error	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	18. What will be the output of the following C code?
	
	`#include <stdio.h>`
	`void main()`
	`{`
	`    char *s = "hello";`
	`    char *n = "cjn";`
	`    char *p = s + n;`
	`    printf("%c\t%c", *p, s[1]);`

	`}` a) h e b) Compile time error c) c o d) h n	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	19. What will be the output of the following C code?
	
	`#include <stdio.h>`
	`int main()`
	`{`
	`    void *p;`
	`    int a[4] = {1, 2, 3, 4};`
	`    p = &a[3];`
	`    int *ptr = &a[2];`
	`    int n = (int*)p - ptr;`
	`    printf("%d\n", n);`
	`}`
	
	a) 1

	b) Compile time error c) Segmentation fault d) 4	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	20. What will be the output of the following C code?
	
	`#include <stdio.h>`
	`void main()`
	`{`
	`    int k = 5;`
	`    int *p = &k;`
	`    int **m  = &p;`
	`    **m = 6;`
	`    printf("%d\n", k);`
	`}`
	
	a) 5
	
	b) Compile time error
	
	c) 6

	d) Junk	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	1. Integer m=10, n= 35, p=5, d=6 Comment about the output of the given two statements Print m*n + p/d Print p/d + m*n A. Differ by 10 B. Same C. Differ by 20 D. Differ due to left and right precedence	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	m*n + p/d = 10*35 + 5/6 = 350 + 0 = 350 p/d + m*n = 5/6 + 10*35 = 0 + 350 = 350 Both gives the same answer	
Marks	1	

Question	2. The primary mission of an analyst or systems designer is to:

	A. calculate the Return on Investment B. development of Software Evaluation Tool C. create a Data Flow Diagram D. extract the physical requirements of the users and convert them to software	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	System designers are intended to create software based on the requirements of their clients. They need to extract the physical requirements of the users and convert them to software.	
Marks	1	

Question	3. In a Singly Circular Linked List, how many address fields are there? A. 1+1 B. 2+1

	C. 1 D. 2	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	Always a singly linked list will have a data field and an address field. The last node of the linked list will be pointing the head if it is a circular linked list else it will be pointing to null.	
Marks	1	

Question	4. Select the best suitable answer for the below functions. sizeof() strlen A. sizeof() - Returns size of string including null characters strlen() - Returns size of string excluding null characters B. sizeof() - Returns size of string including null characters strlen() - Returns size of string including null characters C. sizeof() - Returns size of string excluding null characters

	strlen() - Returns size of string excluding null, characters D. sizeof() - Returns size of string excluding null characters strlen() - Returns size of string including null characters	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	sizeof() always gives the total size of any datatype allocated by the compiler. Where the strlen() gives the length of the string i.e number of characters in a string.	
Marks	1	

Question	5. Which of the following options best suits for 'Memory Leak Occurred' A. Resource allocation pending while debugging the code B. Program releases resources allocated in the memory C. Program does not free the memory which is allocated dynamically D. Occurs due to address assignment failure.

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	Whenever we are allocating memory dynamically we have to delete the memory before the execution of the program completes else a memory leak will happen. If a program has memory leaks, then its memory usage is satirically increasing since all systems have a limited amount of memory and memory is costly. Hence it will create problems.	
Marks	1	

Question	6. When we implement stack by using linked list then: A. Insertion of the node is done from the end and deletion from the end B. Insertion of the node is done from beginning and deletion from end C. Insertion of the node is done from beginning and deletion from beginning D. Insertion of the node is done from end and deletion from beginning	
Type	multiple_choice	
Option	C	correct

Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	Whenever you are implementing a stack using Linked list both insertion and deletion should happen at the same end. Here both A and C are correct. But in stack always we will deal the data in the top i.e beginning hence option c is the best answer.	
Marks	1	

Question	7. Match the following:

Index 1	Math functions in C language	Index 2	Output in float data type
A	ceil(3.6)	1	32.000000
B	floor(3.6)	2	3.000000
C	pow(2,5)	3	12.000000
D	abs(-12)	4	25.000000

A. A-5, B-2, C-4, D-3

B. A-2, B-5, C-4, D-3

C. A-5, B-2, C-1, D-6

D. A - 5, B - 2, C - 1, D - 3

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect

Solution		
Marks	1	

Question	8. Which of the following is not a storage class specifier in C? A. auto B. register C. static D. volatile	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	AUTO - AUTOMATIC MEMORY ALLOCATION; REGISTER - ALLOCATES MEMORY IN CPU REGISTER; STATIC - ALLOCATES MEMORY FOR A VARIABLE (LOCAL OR GLOBAL)THAT CAN BE USED IN WHOLE PROGRAM.	
Marks	1	

Question	9. What is the output of the following program?
	# include <stdio.h>
	
	void fun(int x)
	
	{
	
	  x = 30;

	`}`
	
	`int main()`
	
	`{`
	
	`  int y = 20;`
	
	`  fun(y);`
	
	`  printf("%d", y);`
	
	`  return 0;`
	
	`}`
	
	A. 30
	B. 20
	C. Compiler Error
	D. Runtime Error	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	In C, the arguments are always passed by value, unless and until we explicitly pass them by reference. The changes made to the value of the variable inside the function definition does not reflect in the main function in case of pass by value. So in the given program, the value of y remains unchanged even after the function call.	
Marks	1	

Question	10. What is the output of the following program?
	`# include <stdio.h>`
	`void fun(int *ptr)`
	`{`
	`  *ptr = 30;`
	`}`
	`int main()`
	`{`
	`  int y = 20;`
	`  fun(&y);`
	`  printf("%d", y);`
	`  return 0;`
	`}`
	A. 30
	B. 20
	C. Compiler Error
	D. Runtime Error	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	In the function call statement ‘fun(&y)’, the address of the variable 'y' is passed, so that the value of 'y' gets changed using its address. The address passed from the main function gets stored in the pointer variable 'ptr'. In the statement ‘*ptr = 30’, the value at the address 'ptr' gets changed to 30. Since the value is changed with reference to the address, the changes get reflected in the main function also.	
Marks	1	

Question	11. Consider a compiler where int takes 4 bytes, char takes 1 byte and pointer takes 4 bytes.
	#include <stdio.h>
	int main()
	{
	  int arri[] = {1, 2 ,3};
	  int *ptri = arri;
	  char arrc[] = {1, 2 ,3};
	  char *ptrc = arrc;
	  printf("sizeof arri[] = %d ", sizeof(arri));
	  printf("sizeof ptri = %d ", sizeof(ptri));
	  printf("sizeof arrc[] = %d ", sizeof(arrc));
	  printf("sizeof ptrc = %d ", sizeof(ptrc));
	  return 0;
	}
	A. sizeof arri[] = 3 sizeof ptri = 4 sizeof arrc[] = 3 sizeof ptrc = 4
	B. sizeof arri[] = 12 sizeof ptri = 4 sizeof arrc[] = 3 sizeof ptrc = 1
	C. sizeof arri[] = 3 sizeof ptri = 4 sizeof arrc[] = 3 sizeof ptrc = 1
	D. sizeof arri[] = 12 sizeof ptri = 4 sizeof arrc[] = 3 sizeof ptrc = 4
	E.	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	The size of an array is the number of elements multiplied by the size of the array type. In the given program, the array 'arri' is of integer type and the number of elements stored in it is 3. Hence, sizeof(arri) is 12 bytes. In C, the size of the pointer is fixed, which is 4 bytes. Next, the array 'arrc' is of	

	character type and the number of elements is 3. Hence, sizeof(arrc) is 3 bytes.	
Marks	1	

Question	12. Assume that float takes 4 bytes, predict the output of the following program.
	`#include <stdio.h>`
	`int main()`
	`{`
	`  float arr[5] = {12.5, 10.0, 13.5, 90.5, 0.5};`
	`  float *ptr1 = &arr[0];`
	`  float *ptr2 = ptr1 + 3;`
	`  printf("%f ", *ptr2);`
	`  printf("%d", ptr2 - ptr1);`
	`  return 0;`
	`}`
	A. 90.500000 3
	B. 90.500000 12
	C. 10.000000 12
	D. 0.500000 3	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	Assume that the base address of the array 'arr' is 1000. So, the value 1000 gets stored in the pointer variable 'ptr1'. In the statement *ptr2 = ptr1 + 3;, the compiler actually does the operation of *ptr2 = ptr1 + (3*size of data type pointed by ptr1);. Hence, ptr2 becomes 1012. The value stored at the	

	address 1012 is 90.5 and the same gets printed using the first printf function. Pointer subtraction is possible only when both the pointers point to the variable of the same data type. When two pointer variables are subtracted, the compiler actually does the operation of (ptr2 - ptr1)/size of data type pointed by the pointer. ptr2 is 1012 and ptr1 is1000. (1012 - 1000)/4 = 3. Hence, the value 3 gets printed using the last printf function.	
Marks	1	

Question	13.
	`#include<stdio.h>`
	`int main()`
	`{`
	`  int arr[] = {10, 20, 30, 40, 50, 60};`
	`  int *ptr1 = arr;`
	`  int *ptr2 = arr + 5;`
	`  printf("Number of elements between two pointer are: %d.", (ptr2 - ptr1));`
	`  printf("Number of bytes between two pointers are: %d", (char*)ptr2 -`
	`(char*) ptr1);`
	`  return 0;`
	`}`
	
	Assume that an int variable takes 4 bytes and a char variable takes 1 byte
	
	A. Number of elements between two pointer are: 5. Number of bytes between two pointers are: 20
	B. Number of elements between two pointer are: 20. Number of bytes between two pointers are: 20
	C. Number of elements between two pointer are: 5. Number of bytes between two pointers are: 5
	D. Compiler Error

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	Assume that the base address of the array 'arr' is 1000. In C, the array name represents the base address of the array. So, the value 1000 gets stored in the pointer variable 'ptr1'. arr + 5 gives the value of 1020 as 1000+(5*4) = 1020. Hence, ptr2 - ptr1 = (1020-1000)/4 = 5. Type Casting a pointer into character returns the value of the pointer itself. Hence, (char*)ptr2 - (char*) ptr1 = 1020 - 1000 = 20.	
Marks	1	

Question	14. What is the output of the above program?
	`#include<stdio.h>`
	`int main()`
	`{`
	`  int a;`
	`  char *x;`
	`  x = (char *) &a;`
	`  a = 512;`
	`  x[0] = 1;`
	`  x[1] = 2;`
	`  printf("%d\n",a);`
	`  return 0;`
	`}`
	A. Machine dependent
	B. 513
	C. 258
	D. Compiler Error

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	Although 513; The output differs from machine to machine. We have two types of machines like little-endian and big-endian. In each of the machines, the way of storing the input value differs. Hence, we cannot tell the exact answer for the program.	
Marks	1	

Question	15. What is the output of the below program?
	`#include<stdio.h>`
	`int main()`
	`{`
	`char *ptr = "PROcoder";`
	`printf("%c\n", *&*&*ptr);`
	`return 0;`
	`}`
	A. Compiler Error
	B. Garbage Value
	C. Runtime Error
	D. P	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	In the concept of a pointer, the operator * is used for dereferencing and the operator & is used to get the address of a variable. We can use them alternatively for n number of times. These operators cancel out the effect of each other when they used one after another. In the given program, 'ptr' is a	

	pointer variable pointing to the first character of the string "PROcoder". Hence, *ptr prints P.	
Marks	1	

Question	16. What is the output of the below program?
	`#include<stdio.h>`
	`void fun(int arr[])`
	`{`
	`  int i;`
	`  int arr_size = sizeof(arr)/sizeof(arr[0]);`
	`  for (i = 0; i < arr_size; i++)`
	`    printf("%d ", arr[i]);`
	`}`
	`int main()`
	`{`
	`  int i;`
	`  int arr[4] = {10, 20 ,30, 40};`
	`  fun(arr);`
	`  return 0;`
	`}`
	A. 10 20 30 40
	B. Machine Dependent
	C. 10 20
	D. Northing	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	In programming, when we pass an array to the function definition, it has to be passed along with its size. If not, the compiler will consider the array as a pointer. In the given program, only the array is passed to the function definition. Hence, 'arr' is considered as a pointer. 'arr_size' will become the ratio between pointer size and integer size. The value of the ratio differs	

	from machine to machine. Hence, we cannot tell the output of the above program.	
Marks	1	

Question	17. The reason for using pointers in a C program is A. Pointers allow different functions to share and modify their local variables. B. To pass large structures so that complete copy of the structure can be avoided. C. Pointers enable complex “linked" data structures like linked lists and binary trees. D. All of the above	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	18. What is the output of the following program?
	`#include <stdio.h>`
	`int main()`
	`{`
	`  int *ptr;`
	`  int x;`
	`  ptr = &x;`
	`  *ptr = 0;`
	`  printf(" x = %d\n", x);`
	`  printf(" *ptr = %d\n", *ptr);`

	` *ptr += 5;`
	` printf(" x = %d\n", x);`
	` printf(" *ptr = %d\n", *ptr);`
	` (*ptr)++;`
	` printf(" x = %d\n", x);`
	` printf(" *ptr = %d\n", *ptr);`
	` return 0;`
	`}`
	
	A. x = 0, *ptr = 0; x = 5, *ptr = 5; x = 6, *ptr = 6
	B. x = garbage value, *ptr = 0; x = garbage value, *ptr = 5; x = garbage value, *ptr = 6
	C. x = 0, *ptr = 0; x = 5,*ptr = 5; x = garbage value, *ptr = garbage value;
	D. x = 0, *ptr = 0; x = 0,*ptr = 0; x = 0, *ptr = 0	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	In the given program, 'ptr' stores the address of the variable 'x'. *ptr = 0 sets the value of x to 0. Hence, the value 0 gets printed in the first two printf functions. In the statement *ptr += 5;, the value of x gets incremented by 5. Hence, the value 5 gets printed in the net two printf functions. In the statement (*ptr)++;, the value of x gets incremented by 1. It becomes 6. Hence, the value 6 gets printed in the last two printf functions.	
Marks	1	

Question	19. What is the output of the following program?
	
	`#include <stdio.h>`
	`int main()`
	`{`
	`  static int i=5;`
	`  if (--i){`
	`    printf("%d ",i);`
	`    main();`
	`  }`
	`}`

	A. 4 3 2 1 B. 1 2 3 4 C. 4 4 4 4 D. 0 0 0 0	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	In the given program, the decremented value of 'i' gets printed first and then the main() function is called recursively. Hence, option a is the right answer.	
Marks	1	

Question	20. What is the output of the following program?
	
	`#include <stdio.h>`
	`int main()`
	`{`
	`  int x = 5;`
	`  int * const ptr = &x;`
	`  ++(*ptr);`
	`  printf("%d", x);`
	`  return 0;`
	`}`
	
	A. Compiler Error
	B. Runtime Error
	C. 6
	D. 5

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	First, let's understand the difference between a constant pointer and a pointer to a constant. int * const ptr is a constant pointer. In the case of a constant pointer, we can change the value at the location pointed by the pointer variable. But, we cannot change the pointer to point to a different location. int const * ptr is a pointer to a constant. In this case, we can change the pointer to point to a different location. But, we cannot change the value at the location pointed by the pointer variable. In the given program, a constant pointer is declared. Hence, we can increment the value of 'x' (the variable pointed by the pointer variable 'ptr').	
Marks	1	

Question	1. What is the output of the following program?
	
	`#include <stdio.h>`
	`int main()`
	`{`
	`  int x = 5;`
	`  int const * ptr = &x;`
	`  ++(*ptr);`
	`  printf("%d", x);`
	`  return 0;`
	`}`
	A. Compiler Error
	B. Runtime Error
	C. 6
	D. 5	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	First, let's understand the difference between a constant pointer and a pointer to a constant. int * const ptr is a constant pointer. In the case of a constant pointer, we can change the value at the location pointed by the pointer variable. But, we cannot change the pointer to point to a different location. int const * ptr is a pointer to a constant. In this case, we can change the pointer to point to a different location. But, we cannot change the value at the location pointed by the pointer variable.	

	In the given program, a pointer to a constant is declared. Hence, we cannot increment the value of 'x' (the variable pointed by the pointer variable 'ptr').	
Marks	1	

Question	2. What is the output of the following program?
	
	`#include<stdio.h>`
	`int main()`
	`{`
	`  typedef static int *i;`
	`  int j;`
	`  i a = &j;`
	`  printf("%d", *a);`
	`  return 0;`
	`}`
	
	A. Runtime Error
	B. 0
	C. Garbage Value
	D. Compiler Error	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	In the given program, the first declaration statement has two storage class specifiers namely 'typedef' and 'static'. As per C standard, we can use only one storage class specifier. Hence, the compiler throws an error.	
Marks	1	

Question	3. What is the output of the following program? `#include<stdio.h>`

	int main()
	{
	  typedef int i;
	  i a = 0;
	  printf("%d", a);
	  return 0;
	}
	A. Compiler Error
	B. Runtime Error
	C. 0
	D. 1	

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	In the given program, the first declaration statement has only one storage class specifier. Hence, the program works fine. The compiler creates a user-defined type 'i' and creates a variable 'a' of type 'i'. Now, the variable 'a' becomes an integer type. When we print 'a', the value 0 gets printed.	
Marks	1	

Question	4. What is the output of the following program?
	
	#include<stdio.h>
	int main()
	{
	  typedef int *i;
	  int j = 10;
	  i *a = &j;
	  printf("%d", **a);
	  return 0;
	}
	A. Compiler Error

	B. Garbage Value C. 10 D. 0	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	In the given program, the variable 'i' is a pointer to an integer type. In order to make the variable 'a' to point to the address of the variable 'j', it is enough if we write as i a = &j;. But, the line i *a = &j; makes 'a' as a double-pointer. However, 'a' will not become a double-pointer, as it is not initialized with the address of a pointer.	
Marks	1	

Question	5. What is the output of the following program?
	
	`#include <stdio.h>`
	`int fun()`
	`{`
	`  static int num = 16;`
	`  return num--;`
	`}`
	`int main()`
	`{`
	`  for(fun(); fun(); fun())`
	`    printf("%d ", fun());`
	`  return 0;`
	`}`
	
	A. Infinite loop
	B. 13 10 7 4 1
	C. 14 11 8 5 2

	D. 15 12 8 5 2	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	First, the function 'fun()' is called using the function call in the initialization field. The value of 'num' becomes 16 (for loop initialization is done). Next, the function call in the condition field is executed. Condition checks for a non-zero value. The function 'fun()' returns the value 15 to the condition filed. So, the execution control goes inside the body of the loop. Now, the function call in the printf function is executed. Hence, the value 14 gets printed. Next, the function call in the increment/decrement field is executed and the value of number becomes 13. Again, the condition is checked (num = 12) and the value 11 gets printed. The above process continues until the value of num becomes 0.	
Marks	1	

Question	6. What is the output of the following program?
	
	`#include <stdio.h>`
	`int main()`
	`{`
	`  int x = 10;`
	`  static int y = x;`
	`  if(x == y)`
	`    printf("Equal");`
	`  else if(x > y)`
	`    printf("Greater");`
	`  else`
	`    printf("Less");`
	`  return 0;`
	`}`
	
	A. Compiler Error
	B. Equal

	C. Greater D. Less

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question

7.

```
#include <stdio.h>
#include<string.h>
int main()
{
  struct site
  { // We can't initialize variables inside a structure -
compile error
    //char name[] = "PROcoder";
    //int no_of_pages = 300;

    int no_of_pages;
    char name[];
  };
  struct site *ptr;
  ptr->no_of_pages = 300;
  strcpy(ptr->name,"PROcoder");
  printf("%d ", ptr->no_of_pages);
  printf("%s", ptr->name);
  getchar();
  return 0;
}
```

	A. 300 PROcoder B. 300 C. Runtime Error D. Compiler Error

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	8.
	struct node
	{
	  int i;
	  float j;
	};
	struct node *s[10];
	The above C declaration define 's' to be ______
	(GATE CS 2000)
	
	A. An array, each element of which is a pointer to a structure of type node
	B. A structure of 2 fields, each field being a pointer to an array of 10 elements
	C. A structure of 3 fields: an integer, a float, and an array of 10 elements

	D. An array, each element of which is a structure of type node.	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	In the given program, 's' is an array, where each element is a pointer to a structure of type node.	
Marks	1	

Question	9. Predict the output of the following program
	`#include<stdio.h>`
	`struct st`
	`{`
	`  int x;`
	`  struct st next;`
	`};`
	
	`int main()`
	`{`
	`  struct st temp;`
	`  temp.x = 10;`
	`  temp.next = temp;`
	`  printf("%d", temp.next.x);`
	`  return 0;`
	`}`
	A. Compiler Error
	B. 10
	C. Runtime Error

	D. Garbage Value	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	A structure cannot contain a member of its own type. If so, then it becomes impossible for the compiler to know the size of such structure.	
Marks	1	

Question	10. Predict the output of the following code
	`#include<stdio.h>`
	`struct Point`
	`{`
	`  int x, y, z;`
	`};`
	`int main()`
	`{`
	`  struct Point p1 = {.y = 0, .z = 1, .x = 2};`
	`  printf("%d %d %d", p1.x, p1.y, p1.z);`
	`  return 0;`
	`}`
	A. Compiler Error
	B. 2 0 1
	C. 0 1 2
	D. 2 1 0

Type	multiple_choice

Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	11. The Assumed sizeof an integer and a pointer is 4 byte. Predict the output of the following code.
	
	#include <stdio.h>
	#define R 10
	#define C 20
	int main()
	{
	  int (*p)[R][C];
	  printf("%d", sizeof(*p));
	  getchar();
	  return 0;
	}
	
	A. 200
	B. 4
	C. 800
	D. 80	

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	The output is 10*20*sizeof(int) which is “800″ for compilers with integer size as 4 bytes. When a pointer is dereferenced using *, it gives the type of	

	the object being pointed. In the given program, an array of an array is declared.	
Marks	1	

Question	12. Predict the output of the following code	

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
void fun(char** str_ref)
{
  str_ref++;
}
int main()
{
  char *str = (void *)malloc(100*sizeof(char));
  strcpy(str, "PROcoders");
  fun(&str);
  puts(str);
  free(str);
  return 0;
}
```

A. PROcoders
B. ROcoders
C. Garbage Value
D. Compiler Error

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	In the given program, the variable 'str_ref' is local to the function 'fun()'. When we increment 'str_ref', the value will be changed only inside the function. The changes will not be reflected in the main() function. Hence, the original input string "PROcoders" gets printed as the output.	

Marks	1	

Question	13. ‘ptrdata’ is a pointer to a data type. The expression *ptrdata++ is evaluated as **Hint:** In C, postfix ++ and * operators have the same precedence (priority) and right to left associativity. So ++ works on ptrdata first & them workss on * A. *(ptrdata++) B. (*ptrdata)++ C. *(ptrdata)++ D. Depends on compiler	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	In C, postfix ++ and * operators have the same precedence and right to left associativity. Hence, option a is the correct answer.	
Marks	1	

Question	14. What is the output of the following program
	
	`#include<stdio.h>`
	`int fun(int arr[]) {`
	`  arr = arr+1;`
	`  printf("%d ", arr[0]);`
	`}`
	`int main(void) {`

	```
	  int arr[2] = {10, 20};
	  fun(arr);
	  printf("%d", arr[0]);
	  return 0;
	}
	```
	 A. Compiler Error
	B. 20 10
	C. 20 20
	D. 10 10	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	In C, the array parameter in the function definition will be treated as a pointer. So, the array 'arr' is treated as a pointer in the function definition 'fun()'. The array name 'arr' holds the base address of the array. So, the line arr = arr+1; points to the second memory location. The element at the second index position (20) gets printed using the printf statement inside the function definition. The value at the first index position (10) gets printed using the printf statement inside the main() function.	
Marks	1	

Question	15. What is printed by the following C program?
	```
	#include <stdio.h>
	int f(int x, int *py, int **ppz)
	{
	  int y, z;
	  **ppz += 1;
	```

	```
	  z = **ppz;
	  *py += 2;
	  y = *py;
	  x += 3;
	  return x + y + z;
	}
	void main()
	{
	  int c, *b, **a;
	  c = 4;
	  b = &c;
	  a = &b;
	  printf( "%d", f(c,b,a));
	  getchar();
	}
	```
	 A. 18
	B. 19
	C. 21
	D. 22	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	16. What is the output of the following code?

	#include<stdio.h>
	void fun(int *p)
	{
	  int q = 10;
	  p = &q;
	}
	int main()
	{
	  int r = 20;
	  int *p = &r;
	  fun(p);
	  printf("%d", *p);
	  return 0;
	}
	A. 10
	B. 20
	C. Compiler error
	D. Runtime Error	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	Inside the function 'fun()', the variable 'q' is a copy of the pointer 'p'. So, if we change 'q' to point something else, 'p' remains unchanged. Hence, the value 20 initialized inside the main() function gets printed. If we need to change a local pointer of one function inside another function, we need to pass pointer to a pointer.	
Marks	1	

Question	17. The most appropriate matching for the following pairs X: m=malloc(5); m= NULL; 1: using dangling pointers Y: free(n); n->value=5; 2: using uninitialized pointers Z: char *p; *p = ’a’; 3. lost memory is: A. X-1 Y-3 Z-2 B. X-2 Y-1 Z-3 C. X-3 Y-2 Z-1 D. X-3 Y-1 Z-2	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	X -> m=malloc(5); m= NULL; In the above statement, a pointer is assigned to NULL without freeing memory. So, it shows a clear example of a memory leak. Y -> free(n); n->value=5; In the above statement, we are trying to retrieve the value of 'n' after freeing it. So, it is an example of a dangling pointer. Z -> char *p; *p = ’a’; In the above statement, we are using uninitialized pointers.	
Marks	1	

Question	18. Consider the following three C functions:
	
	[PI]
	//Hint: variable 'x' local variable will be stored in a stack..
	int * g (void)
	{
	  int x= 10;
	  return (&x);
	}
	
	[P2]
	//Hint: variable 'px' is being assigned a value without allocating memory to
	// it.
	int * g (void)
	{
	  int * px;
	  *px= 10;
	  return px;
	}
	
	[P3]
	//Hint: px exists on the heap. Its existence will remain in memory
	//even after the return of g()
	
	int *g (void)
	{
	  int *px;
	  px = (int *) malloc (sizeof(int));
	  *px= 10;
	  return px;
	}

	Which of the above three functions are likely to cause problems with pointers? A. Only P3 B. Only P1 and P3 C. Only P1 and P2 D. P1, P2 and P3	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	In P1, the variable 'x' is a local variable to g(). We all know that a local variable will be stored in a stack. So, &x will become invalid once g() has returned as 'x' is stored in the stack. In P2, the pointer variable 'px' is being assigned a value without allocating memory to it. Hence, p2 will not execute correctly. P3 executes perfectly fine. Memory is allocated to the pointer variable 'px' using malloc(). So, px exists on the heap. Its existence will remain in memory even after the return of g(). Hence, option c is the correct answer.	
Marks	1	

Question	19. Which of the following statement(s) is/are true Hint: descriptive statements have many facts

	A. calloc() allocates contiguous memory and also initializes the allocated memory to zero, while memory allocated using malloc() has random data. Both malloc() and calloc() return 'void *' pointer. B. malloc() and memset() can be used to get the same effect as calloc(). C. calloc() takes two arguments, but malloc takes only 1 argument. D. All of the above	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	All the given statements are True in C. ‘malloc’ does not initialize the allocated memory. If we try to access the content of memory block(before initializing), we’ll get segmentation fault error(or maybe garbage values).‘malloc’ takes one argument which is the size of each block and ‘calloc’ takes two arguments namely number of blocks to be allocated and size of each block.	
Marks	1	

Question	20. Which languages necessarily need heap allocation in the run time environment? A. Those that support recursion B. Those that use dynamic scoping C. Those that use global variables D. Those that allow dynamic data structures

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	Allocating memory during run time is called dynamic memory allocation. Memory allocated by a user during run time has to be freed by the user only. So, the languages that are capable of creating data structure during run time need memory allocation in heap. Note: The name heap does not represent heap data structure. It is a pile of memory space available for the programmers to allocate and deallocate.	
Marks	1	

Test-14 CRT C Loops Arrays & Controls

Question	1. Predict the output of the following program
	`#include <stdio.h>`
	`int main()`
	`{`
	`  int i = 1024;`
	`  for (; i; i >>= 1)`
	`    printf("PROcoder");`
	`  return 0;`
	`}`
	A. 10
	B. 11
	C. Infinite
	D. The program will show compile-time error	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	For the first iteration of the for loop, the condition is true and hence the string "PROcoder" gets printed for the first time. The increment/decrement part of the given for loop contains i >>= 1, which can be written as i = i >>1. The given shift operator is the right shift operator. Hence, the value of 'i' will be right-shifted by one position. Again the condition will be checked. If the condition is true, the string gets printed. This process happens until the shift operation results in the value 0. The binary representation of the value 1024 is 10000000000. There will be 10 right shits (10 iterations) for the binary representation to reach the value	

	0. So, the total number of iterations is 10 + the first iteration. Hence, the string "PROcoder" will get printed 11 times.	
Marks	1	

Question	2. How many times PROcoder is printed in the above program?
	Hint: Given while condition is false.
	
	`#include <stdio.h>`
	`#define PRINT(i, limit) do \`
	`        { \`
	`         if (i++ < limit) \`
	`         { \`
	`           printf("PROcoder\n"); \`
	`           continue; \`
	`         } \`
	`        }while(0);`
	`int main()`
	`{`
	`  int i = 0;`
	`  PRINT(i, 3);`
	`  return 0;`
	`}`
	
	A. 1
	B. 3
	C. 4
	D. Compile-time erroR	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect

Solution	In the do-while loop, the body of a loop is executed at least once. After the body is executed, the while condition will be checked. So, in the given program, the string "PROcoder" gets printed for the first time. After printing, the while condition is checked. The given 'while' condition is false. Hence, the loop gets terminated.	
Marks	1	

Question	3. Predict the output of the following program
	Tip: switch expression contains an integer value and the case labels contain character value
	
	`#include <stdio.h>`
	`int main()`
	`{`
	`  int i = 0;`
	`  switch (i)`
	`  {`
	`    case '0': printf("PRO");`
	`           break;`
	`    case '1': printf("coder");`
	`           break;`
	`    default: printf("PROcoder");`
	`  }`
	`  return 0;`
	`}`
	A. PRO
	B. coder
	C. PROcoder
	D. Compile-time error	

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect

Solution	The switch expression is compared with the values of each case label. If there is a match, the corresponding statements after the matching case label get executed. If there is no match, the default statements get executed. In the given program, the switch expression contains an integer value and the case labels contain character value. So, there is no match between switch expression and case values. Hence, the default value "PROcoder" gets printed.	
Marks	1	

Question	4. Predict the output of the following program
	`#include <stdio.h>`
	`int main()`
	`{`
	`  int i;`
	`  if (printf("0"))`
	`    i = 3;`
	`  else`
	`    i = 5;`
	`  printf("%d", i);`
	`  return 0;`
	`}`
	A. 3
	B. 5
	C. 03
	D. 05	

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect

Solution	Firstly, the 'printf' statement inside the if condition gets executed. Hence, the value 0 gets printed. Since the if condition is true, the variable 'i' gets initialized with the value 3 and the same will be printed using the last printf statement. If the if condition is if(!printf("0")), the variable 'i' gets initialized with the value 5.	
Marks	1	

Question	5.
	#include <stdio.h>
	int i;
	int main()
	{
	  if (i);
	  else
	    printf("Else");
	  return 0;
	}
	What is correct about the above program?
	A. if block is executed.
	B. else block is executed.
	C. It is unpredictable as i is not initialized.
	D. Error: misplaced else	

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	In the given program, the variable 'i' is declared globally. In c, global value has zero as a default value. Hence, the if condition becomes false and else block gets executed. Note: if (i); and if (i) {} are equivalent.	

Marks	1	

Question	6. Predict the output of the following program
	`#include<stdio.h>`
	`int main()`
	`{`
	`  int n;`
	`  for (n = 9; n!=0; n--)`
	`    printf("n = %d", n--);`
	`  return 0;`
	`}`
	A. 9 7 5 3 1
	B. 9 8 7 6 5 4 3 2 1
	C. Infinite Loop
	D. 9 7 5 3	

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	The value of 'n' changes like 7, 5, 3, 1, -1, -3, -5, -7, -9, -11, etc. It will never become 0. Hence, the program executes for infinite times.	
Marks	1	

Question	7. Predict the output of the following program
	TIP: do-while loop gets executed even when the value of c = 0. But, that should not happen.
	
	`#include <stdio.h>`
	`int main()`
	`{`
	`  int c = 5, no = 1000;`
	`  do {`
	`    no /= c;`
	`  } while(c--);`

	`  printf ("%d\n", no);`
	`  return 0;`
	`}`
	
	A. 1
	B. Runtime Error
	C. 0
	D. Compiler Error	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	The statements inside the do-while loop get executed even when the value of c = 0. But, that should not happen. Hence, the program fails during run time.	
Marks	1	

Question	8. Predict the output of the following program
	`# include <stdio.h>`
	`int main()`
	`{`
	`  int i = 0;`
	`  for (i=0; i<20; i++)`
	`   {`
	`    switch(i)`
	`     {`
	`      case 0:`
	`        i += 5;`
	`      case 1:`
	`        i += 2;`
	`      case 5:`
	`        i += 5;`
	`      default:`
	`        i += 4;`
	`        break;`
	`     }`

	`printf("%d ", i);`
	`}`
	`return 0;`
	`}`
	A. 5 10 15 20
	B. 7 12 17 22
	C. 16 21
	D. Compiler Error	

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	9. Predict the output of the below program
	`#include <stdio.h>`
	`int main()`
	`{`
	`  int arr[5];`
	`  // Assume the base address of arr as 2000 and size of an integer as 32 bit`
	`  arr++;`
	`  printf("%u", arr);`
	`  return 0;`
	`}`
	A. 2002
	B. 2004
	C. 2020

	D. lvalue required	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	In C, array name is a constant. It is not a modifiable lvalue. We can add any value to the array name. But, we cannot modify it.	
Marks	1	

Question	10. Predict the output of the below program
	Tip: An array name without & points to 1st array element. &arrayname + 1 means 1st element address + the size of whole array.
	
	#include <stdio.h>
	int main()
	{
	 int arr[5];
	 // Assume the base address of arr as 2000 and size of integer as 32 bit
	 printf("%u %u", arr + 1, &arr + 1);
	 return 0;
	}
	A. 2004 2020
	B. 2004 2004
	C. 2004 Garbage value
	D. The program fails to compile because Address-of operator cannot be used with array name	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect

Option	D	incorrect
Solution	In C, an array name gives the address of the first array element. Here, we don't need to use & for array name. When we add the value 1 to the array name, the address of the first array element gets added with the size of the array type. In the given program, array 'arr' is of type integer and the address of the first array element is 2000. Hence, arr +1 prints the value 2004. When we use &arr + 1, it means (address of the first array element + the total size of the array). Here, the array is declared with 5 integer elements. Hence, the total size of the array is 20. So, &arr + 1 prints 2020.	
Marks	1	

Question	11. Predict the output of the below program
	`# include <stdio.h>`
	`void print(int arr[])`
	`{`
	`  int n = sizeof(arr)/sizeof(arr[0]);`
	`  int i;`
	`  for (i = 0; i < n; i++)`
	`    printf("%d ", arr[i]);`
	`}`
	`int main()`
	`{`
	`  int arr[] = {1, 2, 3, 4, 5, 6, 7, 8};`

	print(arr);
	return 0;
	}
	A. 1, 2, 3, 4, 5, 6, 7, 8
	B. Compiler Error
	C. 1 2
	D. Run Time Error	

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	In the given program, the function 'fun()' receives an array 'arr[]' and finds the number of array elements using 'sizeof' operator. In C, the array parameters passed to the function definition are treated as pointers. Note: According to the compiler 'void print(int arr[])' and 'void print(int *arr)' are the same. So, the expression inside the function 'sizeof(arr)/sizeof(arr[0])' becomes 'sizeof(int *)/sizeof(int)', which results in the value 2, as the size of int* is 8 bytes and int is 4 bytes. Hence, the for loop inside the function definition is executed twice irrespective of the size of the array.	
Marks	1	

Question	12. Predict the output of the below program #include<stdio.h> int main()

	{
	  int a[] = {1, 2, 3, 4, 5, 6};
	  int *ptr = (int*)(&a+1);
	  printf("%d ", *(ptr-1));
	  return 0;
	}
	A. 1
	B. 2
	C. 6
	D. Runtime Error	

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	&a+1 gives the base address of a[] + sizeof(a). The resultant value is type casted into int*. Note: Here, int *ptr = (int*)(&a + 1); is equivalent to int *ptr = &a[6]; Hence, 'ptr' points to the memory address next to the memory address of the value 6. So, ptr - 1 points to the memory address of the value 6. Hence, the output is 6.	
Marks	1	

Question	13. Consider the following C-function in which a[n] and b[m] are two sorted integer arrays and c[n + m] be another integer array.
	void xyz(int a[], int b [], int c[])
	{
	  int i, j, k;
	  i = j = k = O;
	  while ((i<n) && (j<m))

	if (a[i] < b[j]) c[k++] = a[i++];
	else c[k++] = b[j++];
	}
	Which of the following condition(s) hold(s) after the termination of the while loop?
	
	(i) j < m, k = n+j-1, and a[n-1] < b[j] if i = n
	(ii) i < n, k = m+i-1, and b[m-1] <= a[i] if j = m
	
	A. only (i)
	B. only (ii)
	C. either (i) or (ii) but not both
	D. neither (i) nor (ii)	

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	The function xyz() is similar to merge() of mergeSort(). The condition (i) is true if the last inserted element in c[] is from a[] and condition (ii) is true if the last inserted element is from b[].	
Marks	1	

Question	14. Which of the following is true about arrays in C. A. For every type T, there can be an array of T. B. For every type T except void and function type, there can be an array of T. C. When an array is passed to a function, C compiler creates a copy of array. D. 2D arrays are stored in column major form

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	In C, we cannot have an array of void type and function type. Hence, option a is false. When an array is passed to a function, it gets passed as a pointer. Hence, option c is also false. In C, 2D arrays are stored in a row-major form. Hence, option d is also false.	
Marks	1	

Question	15. Predict the output of the following program
	`#include <stdio.h>`
	`int main()`
	`{`
	`  int i;`
	`  int arr[5] = {1};`
	`  for (i = 0; i < 5; i++)`
	`  printf("%d ", arr[i]);`
	`  return 0;`
	`}`
	A. 1 followed by four garbage values
	B. 1 0 0 0 0
	C. 1 1 1 1 1
	D. 0 0 0 0 0	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect

Option	D	incorrect
Solution	{1} initializes only the 1st element of the array. The remaining 4 elements would be by default 0.	
Marks	1	

Question	16. Predict the output of the below program:	

```
#include <stdio.h>
#define SIZE(arr) sizeof(arr) / sizeof(*arr);
void fun(int* arr, int n)
{
  int i;
  *arr += *(arr + n - 1) += 10;
}
void printArr(int* arr, int n)
{
  int i;
  for(i = 0; i < n; ++i)
    printf("%d ", arr[i]);
}
int main()
{
  int arr[] = {10, 20, 30};
  int size = SIZE(arr);
  fun(arr, size);
  printArr(arr, size);
  return 0;
}
```

A. 20 30 40
B. 20 20 40
C. 50 20 40
D. Compile-time error

Type	multiple_choice	
Option	C	correct
Option	A	incorrect

Option	B	incorrect
Option	D	incorrect
Solution	In the given program, the original values of the array 'arr' get changed in the function 'fun' using the expression *arr += *(arr + n - 1) += 10; and the resulting values get printed in the function 'printArr'. First, the value 10 is added to the last element of the array. The result is then added to the first element of the array.	
Marks	1	

Question	17. In the context of the below program snippet, pick the best answer.
	
	`#include "stdio.h"`
	`int arr[10][10][10];`
	`int main()`
	`{`
	`  arr[5][5][5] = 123;`
	`  return 0;`
	`}`
	
	Which of the given printf statement(s) would be able to print arr[5][5][5]
	
	(i) `printf("%d",arr[5][5][5]);`
	(ii) `printf("%d",*(*(*(arr+5)+5)+5));`
	(iii) `printf("%d",(*(*(arr+5)+5))[5]);`
	(iv) `printf("%d",*((*(arr+5))[5]+5));`
	
	A. only (i) would compile and print 123.
	B. both (i) and (ii) would compile and both would print 123.
	C. only (i), (ii) and (iii) would compile but only (i) and (ii) would print 123.
	D. all (i), (ii), (iii) and (iv) would compile and all would print 123.

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	18. Choose the best statement with respect to following three program snippets.
	
	/*Program Snippet 1 with for loop*/
	for (i = 0; i < 10; i++)
	{
	/*statement1*/
	continue;
	/*statement2*/
	}
	
	/*Program Snippet 2 with while loop*/
	i = 0;
	while (i < 10)
	{
	/*statement1*/
	continue;
	/*statement2*/
	i++;
	}
	
	/*Program Snippet 3 with do-while loop*/
	i = 0;
	do
	{
	/*statement1*/
	continue;
	/*statement2*/
	i++;
	}while (i < 10);

	A. All the loops are equivalent i.e. any of the three can be chosen and they all will perform exactly same. B. continue can't be used with all the three loops in C. C. After hitting the continue; statement in all the loops, the next expression to be executed would be controlling expression (i.e. i < 10) in all the 3 loops. D. None of the above is correct.	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	19. With respect to following for loops in C, pick the best statement
	Assume that there is a prior declaration of 'i' in all cases
	Hint: Order of execution is Initialization, Condition & then Incr/Decr. Position can be interchangeable.
	
	for (i < 10; i = 0 ; i++) // (i)
	for (i < 10; i++ ; i = 0) // (ii)
	for (i = 0; i < 10 ; i++) // (iii)
	for (i = 0; i++ ; i < 10) // (iv)
	for (i++; i = 0 ; i < 10) // (v)

	for (i++; i < 0 ; i = 10) // (vi) A. All the above “for” loops would compile successfully. B. All the above “for” loops would compile successfully. Except (iii), the behaviour of all the other “for” loops depend on compiler implementation. C. Only (iii) would compile successfully. D. Only (iii) and (iv) would compile successfully.	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	All the above given for loops are valid. A for loop has three fields namely initialization, condition, increment/decrement. First, the initialization field gets executed followed by the condition filed. If the condition is true, the execution control goes to the body of the for loop. At last, the increment/decrement field will be executed. These fields can be interchanged in the position. However, they will be executed in the above-mentioned order.	
Marks	1	

Question	20. With respect to following for loops in C, pick the best statement. Assume that there is a prior declaration of 'i' in all cases
	
	Tip: fields are optional but semicolon is a must in case of a for loop.
	
	for (i = 0; i < 10 ; i++) // (i)
	for (; i < 10 ; i++) // (ii)
	for (i = 0; ; i++) // (iii)

	for (i = 0; i < 10 ;) // (iv)
	for (; ;) // (v)
	
	A. Only (i) and (v) would compile successfully. Also (v) can be used as infinite loop.
	B. Only (i) would compile successfully.
	C. All would compile successfully but behavior of (ii), (iii) and (iv) would depend on compiler.
	D. All would compile successfully.	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	All the given for loops are valid. A for loop without any one of the fields is valid. But, a for loop without a semicolon is not valid. The semicolon is a must in case of a for loop.	
Marks	1	

Test-15 CRT C Misc

Question	1. Heap allocation is required for the languages A. that support recursion B. that support dynamic data structures C. that use dynamic scope rules D. none of the abovE	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	2. Assume the following C variable declaration int *A [10], B [10][10]; Of the following expressions I. A [2] II. A [2][3] III. B [1] IV. B [2][3] which will not give compile-time errors if used as left-hand sides of assignment statements in a C program? A. I, II and IV only B. II, III and IV only C. II and IV only

	D. IV only	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	Here, B is a 2D array. But, we are accessing only one index. Hence, the compiler throws an error.	
Marks	1	

Question	3. Which of the followings is correct for a function definition along with storage-class specifier in C language? Hint: Only the memory allocation for a variable in CPU can be declared as function parameter. A. int fun(auto int arg) B. int fun(static int arg) C. int fun(register int arg) D. All of the above	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	In C, the only storage-class specifier that can be used in the parameter declaration is 'register'. Hence, option c is the correct answer.	
Marks	1	

Question	4. In a C program snippet, the followings are used for the definition of Integer variables?
	Hint: Implicit declaration of datatype.
	
	`signed s;`
	`unsigned u;`
	`long l;`
	`long long ll;`
	
	Pick the best statement for these.
	
	A. All of the above variable definitions are incorrect because basic data type int is missing.
	B. All of the above variable definitions are correct because int is implicitly assumed in all of these.
	C. Only “long l;” and “long long ll;” are valid definitions of variables.
	D. Only “unsigned u;” is valid definition of variable.	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	In C, 'signed', 'unsigned', 'long' are called type specifiers. The data type 'int' is assumed in all of the three implicitly. Hence, option b is the correct answer.	
Marks	1	

Question	5. Pick the correct statement for const and volatile. Tip: ‘const’ variable cannot be changed; ‘volatile’ variable can be changed.

	A. const is the opposite of volatile and vice versa. B. const and volatile can't be used for struct and union. C. const and volatile can't be used for typedef. D. const and volatile are independent i.e. it's possible that a variable is defined as both const and volatile.	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	In C, the keywords 'const' and 'volatile' are type qualifiers and these two are independent. The value of a variable declared with the 'const' keyword cannot be changed. The value of a variable declared with the 'volatile' keyword can be changed. These type qualifiers can be applied for struct, union, enum, and typedef.	
Marks	1	

Question	6. In the below statement, ptr1 and ptr2 are uninitialized pointers to int i.e. they are pointing to some random address that may or may not be valid address. `int* ptr1, ptr2;` A. TRUE B. FALSE	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect

Option	C	incorrect
Option	D	incorrect
Solution	In the given declaration, the asterisk symbol (*) is placed before ptr1 and not before ptr2. Here, both ptr1 and ptr2 are uninitialized variables. But, ptr1 is a pointer to int and ptr2 is a variable of type int. Hence, the given statement is FALSE.	
Marks	1	

Question	7. In a C file (say sourcefile1.c), an array is defined as follows. Here, we don't need to mention arrary arr size explicitly in [] because the size would be determined by the number of elements used in the initialization.
	`int arr[] = {1,2,3,4,5};`
	
	In another C file (say sourcefile2.c), the same array is declared for usage as follows:
	`extern int arr[];`
	
	In sourcefile2.c, we can use sizeof() on arr to find out the actual size of arr.
	A. TRUE
	B. FALSE	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	In C, the sizeof() operator works only at the compile time. So, we cannot use the sizeof() operator on 'arr' in sourcefile2.c, as 'arr' is an incomplete type (size of the array not mentioned at compile time). But, the array 'arr' in sourcefile1.c is a complete type, as the size of the array is determined at the compiler time because the array elements are initialized at the compile time.	
Marks	1	

Question	8. In the following program snippet, both s1 and s2 would be variables of structure type defined as below and there won't be any compilation issue.
	`typedef struct Student`
	`{int rollno;`
	` int total;`
	`} Student;`
	
	`Student s1;`
	`struct Student s2;`
	
	A. TRUE
	B. FALSE	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	The given statement is True with respect to the given snippet. In the given snippet, structure name is the same as that of typedef name. Having the same structure name and typedef name would not cause any issue. Note: s1 is defined using typedef name and s2 is defined using structure name.	
Marks	1	

Question	9. Pick the best statement for the following program.
	`#include "stdio.h"`
	`int foo(int a){`
	` printf("%d",a);`
	` return 0;`
	`}`
	
	`int main(){`
	` foo;`
	` return 0;`
	`}`

	A. It'll result in compile error because foo is used without parentheses. B. No compile error and some garbage value would be passed to foo function. This would make foo to be executed with output "garbage integer". C. No compile error but foo function wouldn't be executed. The program wouldn't print anything. D. No compile error and ZERO (i.e. 0) would be passed to foo function. This would make foo to be executed with output 0.	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	In C, if a function name is used without parentheses, the reference to the function name generates a pointer to the function, which is then discarded. Hence option c is the correct answer.	
Marks	1	

Question	10. In C, 1D array of int can be defined as follows and both are correct. int array1D[4] = {1,2,3,4}; int array1D[] = {1,2,3,4}; But given the following definitions (along-with initialization) of 2D arrays int array2D[2][4] = {1,2,3,4,5,6,7,8}; /* (i) */int array2D[][4] = {1,2,3,4,5,6,7,8}; /* (ii) */int array2D[2][] = {1,2,3,4,5,6,7,8}; /* (iii) */int array2D[][] = {1,2,3,4,5,6,7,8}; /* (iv) */ Pick the correct statements.

	A. Only (i) is correct. B. Only (i) and (ii) are correct. C. Only (i), (ii) and (iii) are correct. D. All (i), (ii), (iii) and (iv) are correct.

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	In C, a 2D array of int is a 1D array of array of int and it should be declared with the column size. Here, (iii) and (iv) are defined without column size. Hence, option b is the correct answer.	
Marks	1	

Question

11. Pick the best statement for the below program:

```
#include "stdio.h"void fun(int n){
  int idx;
  int arr1[n] = {0};
  int arr2[n];

  for (idx=0; idx<n; idx++)
      arr2[idx] = 0;
}

int main(){
  fun(4);
  return 0;
}
```

A. Definition of both arr1 and arr2 is incorrect because variable is used to specify the size of array. That’s why compile error.

	B. Apart from definition of arr1 arr2, initialization of arr1 is also incorrect. arr1 can't be initialized due to its size being specified as variable. That's why compile error. C. Initialization of arr1 is incorrect. arr1 can't be initialized due to its size being specified as variable. That's why compile error. D. No compile error. The program would define and initializes both arrays to ZERO.	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	In C, when an array is initialized at the time of declaration, it's size cannot be mentioned as a variable. Hence, the declaration of arr1 is incorrect. But, the declaration of arr2 is correct, as arr2 is not initialized at the time of declaration.	
Marks	1	

Question	12. Pick the best statement for the below program:
	`#include "stdio.h"int size = 4;`
	`int arr[size];`
	
	`int main(){`
	` if(arr[0])`
	`  printf("Initialized to ZERO");`
	` elseprintf("Not initialized to ZERO");`
	
	` return 0;`

	} A. No compile error and it'll print "Initialized to ZERO". B. No compile error and it'll print "Not initialized to ZERO". C. Compile error because size of arr has been defined using variable outside any function. D. No compile error and it'll print either "Initialized to ZERO" or "Not initialized to ZERO" depending on what value is present at arr[0] at a particular run of the program.

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	In C, an array whose size is specified as a variable cannot be declared outside any function. In the given program, the array 'arr' is declared outside the main function with the variable 'size' as its size. Hence, option c is the correct answer.	
Marks	1	

Question	13. Consider the following C program:
	 `#include <stdio.h>`
	 `typedef struct`
	 `{`
	 `    char *a;`
	 `    char *b;`
	 `} t;`
	 `void f1(t s);`
	 `void f2(t *p);`

	```
	main()
	{
	    static t s = {"A", "B"};
	    printf ("%s %s\n", s.a, s.b);
	    f1(s);
	    printf ("%s %s\n", s.a, s.b);
	    f2(&s);
	}
	void f1(t s)
	{
	    s.a = "U";
	    s.b = "V";
	    printf ("%s %s\n", s.a, s.b);
	    return;
	}
	void f2(t *p)
	{
	    p -> a  = "V";
	    p -> b = "W";
	    printf("%s %s\n", p -> a, p -> b);
	    return;
	}
	```
	 What is the output generated by the program ?
	
	A. A B U V V W V W
	B. A B U V A B V W
	C. A B U V U V V W
	D. A B U V V W U V	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	14. Consider the code fragment written in C below : void f (int n){

	```
	  if (n <=1)  {
	   printf ("%d", n);
	  }
	  else {
	   f (n/2);
	   printf ("%d", n%2);
	  }
	}
	```
	What does f(173) print?
	
	A. 010110101
	B. 010101101
	C. 10110101
	D. 10101101	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	The given program is used to convert a decimal number to a binary number. The binary equivalent of 173 is 10101101. Hence, option d is the correct answer.	
Marks	1	

Question	15.
	```
	#include <stdio.h>
	            Int main()

	{

	                       int n,ch;

	                     for(n=7;n!=0;n--)

	                       {

	                         printf("n=%d",n--)
	```

	`ch=getchar();`
	`}`
	`Return 0;`
	`}`
	A. Infinite loop
	B. Compilation error
	C. Numbers 7 to 0 in descending order
	D. None of the above	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	As there are syntax errors in the program so it will result in the compilation error.	
Marks	1	

Question	16. `#include <stdio.h>`
	`long int fact (int n);`
	`int main ()`
	`{`
	`int n;`
	`printf (“Enter a positive integer: “);`
	`scanf (“%d”, &n);`
	`printf(“factorial of %d = %ld “, n, fact (n));`
	`return 0;`
	`}`
	`long int fact (int n)`

	```
	{

	if (n != 1)

	return n*fact (n-1)

	else

	return 1;

	}
	```
	A. Iteration
	B. Recursion
	C. Dynamic Programming
	D. Divide & Conquer	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	Since in this program we are having a base condition and the termination condition which are the necessary condition for recursion, so this program uses the recursive approach.	
Marks	1	

Question	17. Which are Incorrect options for array ? A. Same type B. Sequential Memory Allocation C. Size of an array can be changed at runtime D. Counting items in array is appropriate	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect

Solution	As we know the size of the array is fixed and the size of the array cannot be increased at runtime.	
Marks	1	

Question	18. **Ques. Select the missing statement?**
	#include<stdio.h>
	long int fact(int n);
	int main()
	{
	\\missing statement
	}
	long int fact(int n)
	{
	if(n>=1)
	return n*fact(n-1);
	else
	return 1;
	}
	
	A) printf(“%ll\n",fact(5));
	B) printf("%u\n",fact(5));
	C) printf("%d\n",fact(5));
	D) printf("%ld\n",fact(5));	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	19. **Program to print fibonacci series of given range such that series last number will be less than or equal to given no.**
	
	Find Incorrect Statement?
	
	#include<stdio.h>
	int main( )
	{
	int n, a=1, b=1, temp;

	```
	  printf("Enter the number");
	  scanf("%d",&n);

	 printf("Fibonacci series is %d %d ", a,b);

	 while(a+b<=n)    //statement-1
	   {
	     temp=a;
	     b=a;     //Statement-2
	     b=a+temp;
	     printf("%d ",b);
	   }
	return 0;
	}
	```
	 A. Statement-1 only
	B. Statement-2 only
	C. Both Statement-1 & 2
	D. None required	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	Correct Statement should be a=b;	
Marks	1	

Question	20. **Predict the output of following code:**
	```
	 main()
	 {
	int a=10,x;
	x= a-- + ++a;
	printf("%d",x);
	 }
	```
	 A) 19
	B) 20
	C) 22
	D) 23

Type	multiple_choice

Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Test-16 C DS & Sorting

Question	1. Eesha wants to incorporate a history feature. When she presses “go back” then it will be able to visit the previous page. What is data type of data structure used? A. Tree B. Queue C. Stack **D.** Array	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	Stack are used in applications like storing browser history when we click back and forward buttons on a page we visit the previous and next pages that we have visited respectively, these all happen with the help of a stack.	
Marks	1	

Question	2. What does the following function do for a given Linked List with first node as *head*? `void fun1(struct node* head)` `{` `if(head == NULL)` `return;` `fun1(head->next);` `printf("%d  ", head->data);`

	} A. Prints all the nodes of the Linked list B. Prints all the nodes of the Linked list in reverse order C. Prints alternate nodes of the Linked List D. Prints alternate nodes in reverse order	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	In the given code snippet, the function 'fun()' is called recursively until the value of the head becomes NULL. When the value of the head becomes NULL, the execution control goes to the place after the function call. In the given program, the execution control goes to the printf statement. Hence, the data stored in the Linked list gets printed in the reverse order.	
Marks	1	

Question	3. Which of the following points is/are true about Linked List data structure when it is compared with array? A. Arrays have better cache locality that can make them better in terms of performance. B. It is easy to insert and delete elements in Linked List. Random access is not allowed in a typical implementation of Linked Lists C. The size of array has to be pre-decided, linked lists can change their size any time. D. All of the above
Type	multiple_choice

Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	All the given statements are True. ● Since the elements of an array are stored in a contiguous manner, a large number of elements will be loaded into the cache upon first access. This helps in accessing array elements easier. But in a Linked list, the nodes are stored in a random manner. So, there may be cache miss and accessing the elements becomes difficult. ● The process of insertion and deletion in a Linked list is easy, as each and every node are connected by links. Whereas in an array, both the processes are difficult, as the array elements are stored in a contiguous memory location. ● The elements in a Linked list are stored in different memory locations. So, they need to be accessed in a linear manner. But, the array elements can be randomly accessed using their memory address. ● An array size has to be decided at the time of the array declaration. Once declared, the array size cannot be changed. But, the size of a Linked list can be changed during the run time. Hence, all the given options are True with respect to a Linked list.	
Marks	1	

Question	4. Consider the following function that takes reference to head of a Doubly Linked List as parameter. Assume that a node of doubly linked list has previous pointer as *prev* and next pointer as *next*.
	`void fun(struct node **head_ref)`
	`{`
	`    struct node *temp = NULL;`
	`    struct node *current = *head_ref;`
	
	`    while (current !=  NULL)`
	`    {`
	`        temp = current->prev;`
	`        current->prev = current->next;`
	`        current->next = temp;`

	`current = current->prev;`
	`}`
	
	`if(temp != NULL )`
	`*head_ref = temp->prev;`
	`}`
	
	Assume that reference of head of following doubly linked list is passed to above function 1 2 3 4 5 6. What should be the modified linked list after the function call?
	A. 2 <--> 1 <--> 4 <--> 3 <--> 6 <-->5
	B. 5 <--> 4 <--> 3 <--> 2 <--> 1 <-->6.
	C. 6 <--> 5 <--> 4 <--> 3 <--> 2 <--> 1.
	D. 6 <--> 5 <--> 4 <--> 3 <--> 1 <--> 2	

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	The given code snippet reverses a doubly linked list. In a doubly-linked list, each node has three fields namely data field, previous field, and next field. The data field holds the data of the current node, the previous field holds the address of the previous node and the next field holds the address of the next node. In order to reverse a doubly-linked list, we need to swap the previous and the next fields of all the nodes, change the previous field of the head node and the next field of the last node.	
Marks	1	

Question	5. Which of the following sorting algorithms can be used to sort a random linked list with minimum time complexity? A. Insertion Sort B. Quick Sort C. Heap Sort

	D. Merge Sort	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	Both Merge sort and Insertion sort can be used for sorting a Linked list. The slow random-access performance of a linked list makes other algorithms such as Quick sort perform poorly and others such as Heap sort completely impossible. Since the worst-case time complexity of Merge Sort is O(nLogn) and Insertion sort is O(n^2), Merge sort is preferred.	
Marks	1	

Question	6. The following function reverse() is supposed to reverse a singly linked list. There is one line missing at the end of the function.
	`/* Link list node */`
	`struct node`
	`{`
	`    int data;`
	`    struct node* next;`
	`};`
	
	`/* head_ref is a double pointer which points to head (or start) pointer`
	`  of linked list */`
	`static void reverse(struct node** head_ref)`
	`{`
	`    struct node* prev   = NULL;`
	`    struct node* current = *head_ref;`
	`    struct node* next;`
	`    while (current != NULL)`
	`    {`
	`        next  = current->next;`
	`        current->next = prev;`
	`        prev = current;`
	`        current = next;`
	`    }`
	`    /*ADD A STATEMENT HERE*/`
	`}`
	
	What should be added in place of "/*ADD A STATEMENT HERE*/", so that the function correctly reverses a linked list.

	A. *head_ref = prev;
	B. *head_ref = current;
	C. *head_ref = next;
	D. *head_ref = NULL;	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	At the end of the while loop, the *prev* pointer points to the last node of the original linked list. We need to change '*head_ref' so that the head pointer now starts pointing to the last node.	
Marks	1	

Question	7. What is the output of the following function for start pointing to the first node of following linked list? 1->2->3->4->5->6
	`void fun(struct node* start)`
	`{`
	`  if(start == NULL)`
	`    return;`
	`  printf("%d  ", start->data);`
	
	`  if(start->next != NULL )`
	`    fun(start->next->next);`
	`  printf("%d  ", start->data);`
	`}`
	A. 1 4 6 6 4 1
	B. 1 3 5 1 3 5
	C. 1 2 3 5
	D. 1 3 5 5 3 1	

Type	multiple_choice	

Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	The given function definition 'fun()' prints alternate nodes of the given Linked List, first from head node to last node, and then from the last node to head node. If the Linked List has an even number of nodes, then the function 'fun()' eliminates the last node.	
Marks	1	

Question	8. The following C function takes a simply-linked list as an input argument. It modifies the list by moving the last element to the front of the list and returns the modified list. Some part of the code is left blank. Choose the correct alternative to replace the blank line.
	`typedef struct node`
	`{`
	`  int value;`
	`  struct node *next;`
	`}Node;`
	
	`Node *move_to_front(Node *head)`
	`{`
	`  Node *p, *q;`
	`  if ((head == NULL: || (head->next == NULL))`
	`    return head;`
	`  q = NULL; p = head;`
	`  while (p-> next !=NULL)`
	`  {`
	`    q = p;`
	`    p = p->next;`
	`  }`
	`  ________________________________`
	`  return head;`
	`}`
	A. q = NULL; p->next = head; head = p;
	B. q->next = NULL; head = p; p->next = head;
	C. head = p; p->next = q; q->next = NULL;
	D. q->next = NULL; p->next = head; head = p;

Type	multiple_choice

Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	When the while loop ends, q contains the address of the second last node and p contains the address of the last node. So, we need to do the following things after the while loop. i) Set next of q as NULL (q->next = NULL). ii) Set next of p as head (p->next = head). iii) Make head as p (head = p) Step (ii) must be performed before step (iii). If we change head first, then we lose track of the head node in the original linked list.	
Marks	1	

Question	9. The following C function takes a single-linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution?
	`struct node`
	`{`
	`  int value;`
	`  struct node *next;`
	`};`
	`void rearrange(struct node *list)`
	`{`
	`  struct node *p, * q;`
	`  int temp;`
	`  if ((!list) || !list->next)`
	`      return;`
	`  p = list;`
	`  q = list->next;`
	`  while(q)`
	`   {`
	`     temp = p->value;`
	`     p->value = q->value;`
	`     q->value = temp;`
	`     p = q->next;`
	`     q = p?p->next:0;`

	```
	 }
	}
	```
	A. 1,2,3,4,5,6,7
	B. 2,1,4,3,6,5,7
	C. 1,3,2,5,4,7,6
	D. 2,3,4,5,6,7,1	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	The given function definition 'rearrange()' swaps the data of each node with its next node. So, the first value 1 gets swapped with the next value 2. Next, the third value gets with its next value 4 and so on. If there is no next value, the swapping operation will not be performed.	
Marks	1	

Question	10. In the worst case, the number of comparisons needed to search a singly linked list of length 'n' for a given element is A. log 2 n B. n/2 C. log 2 n – 1 D. n	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect

Option	C	incorrect
Solution	In the worst case, the search element has to be compared with all the elements of the Linked list. Since the length of the linked list is 'n', the compiler has to traverse through all the 'n' elements to find the match. Hence, the number of comparisons to search in a singly linked list is 'n'.	
Marks	1	

Question	11. Suppose each set is represented as a linked list with elements in arbitrary order. Which of the operations among union, intersection, membership, cardinality will be the slowest? A. union only B. intersection, membership C. membership, cardinality D. union, intersection	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	For getting the intersection of L1 and L2, search for each element of L1 in L2 and print the elements found in L2. Traversing through each and every element in the Linked list is the slowest process. Similarly finding Union is also the slowest process, as we need to process all the nodes of both the Linked lists.	
Marks	1	

Question	12. Suppose a person sees a picture, so he slides the picture left and can see the next picture and when he wants to see the previous

	image so he slides picture right. So in viewing previous or next image, what type of data structure is used in it? A. Stack B. Queue C. Linked list D. Tree	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	Linked list is used for viewing the image when we see the next or previous image. The address of previous or next image save in current image.	
Marks	1	

Question	13. What is time complexity to count the number of elements of a linked list. A. O(n) B. O(log n) C. O(1) D. None of them	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	o count the number of elements, you have to traverse through the entire list, hence complexity is O(n).	
Marks	1	

Question	14. what type of functionality the following code performs
	
	`Public void(Node node)`
	`{`
	`if(size==0)`
	`head==node;`
	`else`
	`{`
	`Node temp,cur;`
	`for(cur = head(temp=cur.getNext())! = Null; cur = temp);`
	`cur.setNext(node);`
	`}`
	`size++;`
	`}`
	
	A. Insertion from beginning
	B. Deletion from beginning
	C. Insertion from end
	D. Deletion from end

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	This function is for the insertion from the last look up the condition. if the head is empty so we need to add a node at starting otherwise we go to the last node and then add the node.	
Marks	1	

Question	15. What is the functionality of the following piece of code
	
	Public int function(int data)
	{
	Node temp = head;
	Int var=0;
	while(temp!=Null)
	{
	if(temp.getData()==data)
	Return var;
	var=var+1;
	Temp = temp.getNext();
	}
	Return Integer.Min_value;
	}
	A. Find the data
	B. Find the data and return position
	C. Find the data and insert new element
	D. Find the data and return data

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	The function is for the search a given data and then return the position, look at the function here nodes are compared with the given data and at last it return the position.	
Marks	1	

Question	16. Which of the following functionality is true for the count the number of the element of the link list.
	
	A. public void insertBegin(Node node)
	{
	node.setNext(head);
	head = node;
	size++;
	}
	
	B. public void insertBegin(Node node)
	{
	head = node;
	node.setNext(head);
	size++;
	}
	
	C. public void insertBegin(Node node)
	{
	Node temp = head.getNext()
	node.setNext(temp);
	head = node;
	size++;
	}
	
	D. public void insertBegin(Node node)
	{
	Node temp = head.getNext()
	node.setNext(temp);
	node = head;

	size++;
	}
	
	A
	B
	C
	D	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	NDone of these	incorrect
Solution	look at the function here the first node that is head store in the node and till we do not get null in the node so we will do size+1.	
Marks	1	

Question	18. In a circular link list A. Forward and backward traversal within the list is permitted B. Nodes are arranged in hierarchical manner C. There is no beginning and no end D. Compounds are all linked together in sequential manner.	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	As its name suggests it is circular in shape and a circle has no starting point as well as ending point so the same goes for the circular linked list.	
Marks	1	

Question	19. A link list contains minimum two field. One of them is data field to store the data so what is the second field? A. It's a pointer to character. B. It's a pointer to integer. C. It's a pointer to node D. None.	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	When a data is needed to be retrieved from the linked list data structure, then there is a need of going through all the nodes, until the required data is not found.	
Marks	1	

Question	20. The concatenation of two linked list takes O(1) time. Which of the following variation of linked list can be performed. A. Singly linked list. B. Doubly linked list C. Circular linked list D. Circular doubly linked list.	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	You can do it in O(1) with any implementation that uses pointers and in which you can get to the last element in one operation. CLL lets you do a back from the head element to get to the end, then you update a few pointers and you\'re done. However, if you just have a member	

	that points at the last element, you can use any other pointer based implementation just as well.	
Marks	1	

Question		
Type	multiple_choice	
Option		correct
Option		incorrect
Option		incorrect
Option	None of these	incorrect
Solution		
Marks	1	

Test-17 CRT C All

Question	1. Predict the output of the following code snippet:
	
	`#include <stdio.h>`
	`int main()`
	`{`
	`  int i=3;`
	`  switch(i)`
	`  {`
	`    case 0: printf("Purple");`
	`    break;`
	`    case 1+1: printf("Blue");`
	`    break;`
	`    case 7/2: printf("Yellow");`
	`    break;`
	`    case 3%2: printf("Black");`
	`    break;`
	`  }`
	`  return 0;`
	`}`
	
	A. Black
	B. 3%2
	C. error
	D. Yellow	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	Initially, the value of i is 3. We perform a switch action, which can only be performed on char or int data types. For any more complex case than 0, the case takes anything up to the colon as a single entity. Therefore, 1+1=2, 7/2=3(quotient) since it is of integer data type, 3%2=1(remainder). Now, there is a break after any case that is acted upon. This takes the user out of the switch once the case is executed so that there is no repetition.	

	Since the value of i is 3, it will go to case 3, i.e., “case 7/2:” and then the switch will break. The simple output for this code would thus be Yellow.	
Marks	1	

Question	2. The range of a 10-bit unsigned integer is: A. 0-1024 B. 0-1000 C. 0-1048 D. 0-1023	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	0-1023 We know that the number of bits here is 10 and for every bit, there are 2 possible values(0 and 1) Since it is an unsigned integer, no bit is used to represent the sign and all 10 bits will represent the magnitude. Now, the total possible number would be 2^n=2^10=1024. Starting from 0, that gives us the range: 0-1023.	
Marks	1	

Question

3. In the following code, identify the line with an error.

```
# include<stdio.h> //line 1
 # include<conio.h> //line 2
 void main() //line 3
 { //line 4
    float num1 = 100.00; // line 5
   { // line 6
      auto float num1 = 675.29; // line 7
    { // line 8
           auto float num2 = 325; //line 9
          printf("\n%f %f", num1); //line 10
          num2++; // line 11
```

	` } // line 12`
	` num1++; // line 13`
	` printf("\n%f %f", num2, num1); // line 14`
	` num1++; // line 15`
	` } // line 16`
	` printf("\n%f", num1); //line 17`
	` } // line 18`
	
	A. Line 2
	
	B. Line 9
	
	C. Line 12
	
	D. Line 10	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	Line 9 Since ‘num2’ is declared locally in an inner block, its scope doesn’t extend outside the block where it is accessed. To fix the code, we must declare ‘num2’ globally to give it global scope.	
Marks	1	

Question	4. What will the output be?
	`#include <stdio.h>`
	`int main()`
	` {`
	`   double num = (0.4, 4.7, 7.4);`
	`   printf("%f", num);`
	`   return 0;`
	` }`
	
	A. 4.7

	B. error C. 0.004 D. 7.400000	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	7.400000 The bracket operator is executed before the assignment operator due to higher precedence. The result will always be from the last expression assigned when evaluated left to right.	
Marks	1	

Question	5. The in-order traversal of a binary tree is M J N H R K S, and its pre-order traversal is H J M N K R S. According to this, what would be the post-order traversal of the binary tree? A. M J N R K S H B. H J K M N R S C. M N J R S K H D. R S K H M N J	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect

Solution	M N J R S K H The binary tree is shown according to the in-order and preorder traversal. We can build the binary tree by considering its inorder transversal and pre-order transversal. The creation is as follows: Pre-order traversal: Root->Left->Right Inorder traversal: Left->Root->Right , Using the mentioned pattern we could construct the binary tree. For great detail refer to this article.	
Marks	1	

Question	6. What will the given code result in?
	
	`#include <stdio.h>`
	`int main()`
	`  {`
	`   int packets;`
	`   for( ; ; packets++)`
	`   {`
	`      printf("%d", packets);`
	`      if(packets++==6)`
	`      break;`
	`   }`
	`}`
	
	A. 0 1 2 3 4 5 6
	
	B. error
	
	C. 0 2 4 6
	
	D. infinite loop	

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution		

Marks	1	

Question	7. What type of output would the following code produce?
	
	`#include <stdio.h>`
	`int main()`
	`{`
	`    int num = 5;`
	`    printf("%p\n", &num);`
	`    return 0;`
	`}`
	
	A. integer
	
	B. boolean
	
	C. error
	
	D. variable address	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	Variable address. Since %p prints a pointer value, the following code will print the variable address.	
Marks	1	

Question	8. Predict the output:
	
	`#include <bits/stdc++.h>`
	`int main()`
	`{`
	`   int a=0;`
	`   if(a++)`
	`   {`

	```
	    std::cout<<"Hello";
	  }
	  else{
	  std::cout<<"Goodbye";
	  }
	  return 0;
	}
	```
	A. Syntax error
	
	B. Runtime error
	
	C. Hello
	
	D. Goodbye	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	People tend to get confused since there is no condition for ‘if’. However, it is straightforward. The value of ‘a’ is initialized as 0. Now, if the result of whatever we write is 0, it is considered false, while every non-zero result will be true. In post-increment, the value is checked before updation. So the condition will be false, and the else condition will print Goodbye.	
Marks	1	

Question	9. What is the syntax for command-line arguments? A. int main(char c, int arg) B. int main(int var, char*argv[]) C. int main(int v, char c) D. int main(char*arv[], int arg[])

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	B. int main(int var, char*argv[]) The syntax for command-line arguments is int main(int var, char*argv[]), where var is a positive integer variable to store the number of arguments passed by users (including the program name). The ‘argv’ or the argument vector, on the other hand, is an array for character pointers and lists all arguments. ‘argv[0] is the program name and all elements till argv[var-1] are command-line arguments.	
Marks	1	

Question	10. Arrays are also known as: A. Identical variables B. Variable collectives C. Similar quantity variables D. Subscripted variables	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	D. Subscripted variables Since all elements in the array can be identified under the same name, with the index value (subscript value), arrays can also be called subscripted variables.	

Marks	1	

Question	11. A user is trying to pop an element from some empty stack. Such a condition is exclusively called: A. Underflow condition B. Overflow condition C. Garbage collection D. Empty collection	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	A. Underflow condition In case a user tries to delete an element from a stack containing no elements at all, the condition is termed an underflow condition.	
Marks	1	

Question	12. A given tree's height starts from 0. Trisha wants to know the number of nodes in that tree. What do you think would be the number of nodes in a tree with 'h' height? A. 2^h B. 0 C. (h^2 – 1) D. (2^h+1 – 1)	
Type	multiple_choice	

Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	D. (2^h+1 – 1) We know from the formula itself that: if h starts from 0, there will be (2^h+1 – 1) nodes; if h starts from 1, there will be (2^h – 1)nodes in the tree.	
Marks	1	

Question	13. What is the time complexity for a merge-sort algorithm? A. O(n) B. O(log n) C. O(n log(n)) D. O(n^2)	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	C. O(n log n) Merge sort divides the array into two halves and then merges both halves in linear time. Therefore the worst, average and best case time complexity will be O(n*Log n).	
Marks	1	

Question	14. How many moves are needed to solve the Tower of Hanoi puzzle? A. 2^(n)-1

	B. 2^(n)+1 C. 2^n D. n^2	
Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	A. 2^(n)-1 In the Tower of Hanoi, for disc 1, we need 1 move. For disc 2, we need 3 moves and for disc 3 we need 5 moves. Hence, for n discs, we need 2^(n)-1 moves.	
Marks	1	

Question	15. What type of value does a dangling pointer point to? A. Points to a null value B. Points to 0 value C. Points to a garbage value D. Both 1 and 2	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	C. Points to a garbage value	

	Unreferenced objects or garbage value is pointed at by a dangling pointer.	
Marks	1	

Question	16. **What is the advantage of recursive approach than an iterative approach?** A. Consumes less memory B. Less code and easy to implement C. Consumes more memory D. More code has to be written	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	Less code and easy to implement	
Marks	1	

Question	17. **Predict the output of following code:**
	Tip: All variables must be declared before assigning & initializing to avoid syntax issues.
	`main()`
	`{`
	`int a=b=c=d=10;`
	`printf("%d,%d,%d,%d",a,b,c,d);`
	`}`
	A) Error
	B) 10,10,10,10
	C) Garbage Value,Garbage Value,Garbage Value,10
	D) Garbage Value,Garbage Value,Garbage Value,Garbage Value	

Type	multiple_choice	
Option	A	correct

Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	18. **Select the missing statement?**
	`#include<stdio.h>`
	`long int fact(int n);`
	`int main()`
	`{`
	`\\missing statement`
	`}`
	`long int fact(int n)`
	`{`
	`if(n>=1)`
	` return n*fact(n-1);`
	`else`
	` return 1;`
	`}`
	
	A) printf(“%ll\n",fact(5));
	B) printf("%u\n",fact(5));
	C) printf("%d\n",fact(5));
	D) printf("%ld\n",fact(5));	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	19. **Which of the following does not require to include math.h header file?** A. pow()

	B. rand() C. sqrt() D. sinh()	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	**20. Predict the output of following code:** Hint: `a++` (post-increment) or `++a` (pre-increment) **Tip:** with a=1, both a++ and ++a print 2 and 2. However, the assignment statements such as b = ++a and b = a++ print 2 and 1 **Question:** **`main()`** **`{`** **`int a=10,x;`** **`x= a-- + ++a;`** **`printf("%d",x);`** **`}`** A) 19 B) 20 C) 22 D) 23	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	Answer is 20 with a=1, both a++ and ++a print 2 and 2. However, the assignment statements such as b = ++a and b = a++ print 2 and 1	

Marks	1	

Question	Arguments that take input by user before running a program are called? a) Main function arguments b) Main arguments c) Command-Line arguments d) Parameterized arguments	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	What is the maximum number of arguments that can be passed in a single function? Hint: parameters are limited to about 2pow8 a) 127 b) 253 c) 361 d) No limits in number of arguments

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	What will be the output of the following C code?
	`#include <stdio.h>`
	`void m(int p, int q)`
	`{`
	`    int temp = p;`
	`    p = q;`
	`    q = temp;`
	`}`
	`void main()`
	`{`
	`    int a = 6, b = 5;`
	`    m(a, b);`
	`    printf("%d %d\n", a, b);`
	`}`
	a) 5 6
	b) 5 5
	c) 6 5
	d) 6 6

Type	multiple_choice

Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution	A function call passes a copy of parameter values. The swap in function is local to function. The function does not affect main variables.	
Marks	1	

Question	What will be the output of the following C code?
	
	`#include <stdio.h>`
	`void main()`
	`{`
	`    char *s = "hello";`
	`    char *p = s;`
	`    printf("%p\t%p", p, s);`
	`}`
	
	a) Different address is printed
	
	b) Same address is printed
	
	c) Run time error
	
	d) Nothing	

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		

Marks	1	

Question	Comment on an array of the void data type. a) It can store any data-type b) It only stores element of similar data type to first element c) It acquires the data type with the highest precision in it d) You cannot have an array of void data type	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int ary[4] = {1, 2, 3, 4};
    int p[4];
    p = ary;
    printf("%d\n", p[1]);
}
```

	a) 1 b) Compile time error c) Undefined behaviour d) 2	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	error: assignment to expression with array type p = ary; instead declare int *p; p = ary;	
Marks	1	

Question	What will be the output of the following C code?
	
	`#include <stdio.h>`
	`void main()`
	`{`
	`    char *s = "hello";`
	`    char *p = s;`
	`    printf("%c\t%c", *p, s[1]);`
	`}`
	
	a) e h
	
	b) Compile time error

	c) h h d) h e	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	What will be the output of the following C code?
	
	`#include <stdio.h>`
	`int main()`
	`{`
	`    void *p;`
	`    int a[4] = {1, 2, 3, 4};`
	`    p = &a[3];`
	`    int *ptr = &a[2];`
	`    int n = (int*)p - ptr;`
	`    printf("%d\n", n);`
	`}`
	
	a) 1
	
	b) Compile time error
	
	c) Segmentation fault
	
	d) 4

Type	multiple_choice

Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	What will be the output of the following C code?
	
	`#include <stdio.h>`
	`int main()`
	`{`
	`    int a[4] = {1, 2, 3, 4};`
	`    int b[4] = {1, 2, 3, 4};`
	`    int n = &b[3] - &a[2];`
	`    printf("%d\n", n);`
	`}`
	
	a) -3
	
	b) 5
	
	c) 4
	
	d) Compile time error	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		

Marks	1	

Question	What will be the output of the following C code?
	
	`#include <stdio.h>`
	`int main()`
	`{`
	`    char str[] = "hello, world";`
	`    str[5] = '.';`
	`    printf("%s\n", str);`
	`    return 0;`
	`}`
	
	a) hello. world
	
	b) hello, world
	
	c) Compile error
	
	d) Segmentation fault	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	Comment on the output of the following C code.

```c
#include <stdio.h>
int main()
{
    char *str = "This" //Line 1
    char *ptr = "Program\n"; //Line 2
    str = ptr; //Line 3
    printf("%s, %s\n", str, ptr); //Line 4
}
```

a) Memory holding “this” is cleared at line 3

b) Memory holding “this” loses its reference at line 3

c) You cannot assign pointer like in Line 3

d) Output will be This, Program

Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question

Output of following program? (Hint: A compiler may choose to evaluate either from left-to-right or right-to-left.)

```c
#include <stdio.h>
int main()
{
    int i = 5;
    printf("%d %d %d", i++, i++, i++);
    return 0;
}
```

A   7 6 5

	B 5 6 7 C 7 7 7 D Compiler Dependent	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	When parameters are passed to a function, the value of every parameter is evaluated before being passed to the function. What is the order of evaluation of parameters - left-to-right or right-to-left? If evaluation order is left-to-right, then output should be 5 6 7 and if the evaluation order is right-to-left, then output should be 7 6 5. Unfortunately, there is no fixed order defined by C standard. A compiler may choose to evaluate either from left-to-right. So the output is compiler dependent.	
Marks	1	

Question	Which of the following is true about return type of functions in C? A Functions can return any type B Functions can return any type except array and functions C Functions can return any type except array, functions and union D Functions can return any type except array, functions, function pointer and union	
Type	multiple_choice	
Option	B	correct
Option	A	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	In C, functions can return any type except arrays and functions.	

	We can get around this limitation by returning pointer to array or pointer to function.	
Marks	1	

Question	Can we use a function as a parameter of another function? [Eg: void wow(int func())]. a) Yes, and we can use the function value conveniently b) Yes, but we call the function again to get the value, not as convenient as in using variable c) No, C does not support it d) This case is compiler dependent	
Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	Which of the following mathematical function requires 2 parameter for successful function call?
	
	a) fmod();
	
	b) div();
	
	c) atan2();

	d) all of the mentioned	
Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution		
Marks	1	

Question	What will be the output of the following C code?
	
	`#include <stdio.h>`
	`void main()`
	`{`
	`    m();`
	`}`
	`void m()`
	`{`
	`    printf("hi");`
	`    m();`
	`}`
	
	a) Compile time error
	
	b) hi
	
	c) Infinite hi
	
	d) Nothing	

Type	multiple_choice	
Option	C	correct
Option	A	incorrect
Option	B	incorrect

Option	D	incorrect
Solution		
Marks	1	

Question	Which of the following statement is correct?
	`double x, y, z;`
	`x = 5.123456;`
	`z= modf(x, *y);`
	a) y stores integer part of x, z returns fractional part of x
	b) y stores integer part of x, z returns integer part of x
	c) y stores fractional part of x, z returns integer part of x
	d) y stores fractional part of x, z returns fractional part of x	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution	double modf(double x, double *y) This function returns the fractional part of x, with the same sign. modf() function breaks the argument value into integer and fraction parts, each of which has the same sign as the argument. It stores the integer part as a double in the object pointed to by y.	
Marks	1	

Question	How can you write a Macro MAX()?
	
	`#include <stdio.h>`
	
	`int main(void)`
	`{`
	`    int x = 10, y = 15;`
	`    float u = 2.0, v = 3.0;`
	`    double s = 5, t = 5;`
	
	`    printf("Max of two integers %d and %d is: %d\n", x, y, MAX(x,y));`
	`    printf("Max of two floats %.2f and %.2f is: %.2f\n", u, v, MAX(u,v));`
	`    printf("Max of two doubles %.2lf and %.2lfis: %lf\n", s, t, MAX(s,t));`
	
	`    return 0;`
	`}`
	
	A int MAX(a,b) { if a>b return a; }
	
	B #macro MAX(a,b) ((a) > (b) ? (a) : (b))
	
	C #include <MAX(a,b) ((a) > (b) ? (a) : (b))>
	
	D #define MAX(a,b) ((a) > (b) ? (a) : (b))	

Type	multiple_choice	
Option	D	correct
Option	A	incorrect
Option	B	incorrect
Option	C	incorrect
Solution	have the statement right below the #include statement Max of two integers 10 and 15 is: 15 Max of two floats 2.00 and 3.00 is: 3.00 Max of two doubles 5.00 and 5.00is: 5.000000	

Marks	1	

Question	According to ANSI specifications which is the correct way of declaring main when it receives command-line arguments?
	
	A.
	int main(int argc, char *argv[])
	
	B.
	int main(argc, argv)
	int argc; char *argv;
	
	C.
	int main()
	{
	    int argc; char *argv;
	}
	
	D. None of above	

Type	multiple_choice	
Option	A	correct
Option	B	incorrect
Option	C	incorrect
Option	D	incorrect
Solution		
Marks	1	

Question	What do the 'c' and 'v' in argv stands for? A. 'c' means argument control 'v' means argument vector B. 'c' means argument count 'v' means argument vertex C. 'c' means argument count 'v' means argument vector D. 'c' means argument configuration 'v' means argument visibility	
Type	multiple_choice	
Option	C	correct

Option	A	incorrect
Option	B	incorrect
Option	D	incorrect
Solution		
Marks	1	