Qs in C Section 1

What is the range of values that can be stored by int datatype in C?

A. -(2^31) to (2^31) - 1

B. -256 to 255

C. -(2^63) to (2^63) - 1

D. 0 to (2^31) - 1

What is the range of values that can be stored by int datatype in C?

A. -(2^31) to (2^31) - 1

B. -256 to 255

C. -(2^63) to (2^63) - 1

D. 0 to (2^31) - 1

Formula -2^n to $+2^n - 1$ (for Signed)

What will be the output of the following code snippet?

```c
#include <stdio.h>
int main() {
    int a = 3, b = 5;
    int t = a;
    a = b;
    b = t;
    printf("%d %d", a, b);
    return 0;
}
```

A. 3 5

B. 3 3

C. 5 5

D. 5 3

What will be the output of the following code snippet?

```c
#include <stdio.h>
int main() {
    int a = 3, b = 5;
    int t = a;
    a = b;
    b = t;
    printf("%d %d", a, b);
    return 0;
}
```

A. 3 5

B. 3 3

C. 5 5

D. 5 3

What is the output of the following code snippet?

```
int main() {
    int sum = 2 + 4 / 2 + 6 * 2;
    printf("%d", sum);
    return 0;
}
```

A. 2

B. 15

C. 16

D. 18

Following the BEDMAS (BODMAS) rule
B - Brackets
E - Exponent (or O - Of Power / Root)
D, M - Division, Multiplication (L to R)
A, S - Addition, Subtraction (L to R)

Which of the following is not a storage class specifier in C?

A. volatile

B. extern

C. typedef

D. static

Which of the following is not a storage class specifier in C?

A. volatile

B. extern

C. typedef

D. static

CPU Optimization
Normal variable values get transferred to CPU Registers (L1, L2, L3 Cache)

Volatile tells CPU not to Optimize
Because the value of variable does change frequently

Which of the following is the proper syntax for declaring macros in C?

A. #define long long ll
B. #define ll long long
C. #define ll
D. #define long long

Which of the following is the proper syntax for declaring macros in C?

A. #define long long ll

B. #define ll long long

C. #define ll

D. #define long long

To declare a variable x of long long type: (not recommended)

```
#define ll long long

int main ()
{
ll x;
return 0;
}
```

What is the size of the int data type (in bytes) in C?

A. 4

B. 8

C. 2

D. 1

What is the size of the int data type (in bytes) in C?

A. 4

B. 8

C. 2

D. 1

Which of the following are not standard header files in C?

A. stdio.h

B. stdlib.h

C. conio.h

D. None of the above.

Which of the following are not standard header files in C?

A. stdio.h

B. stdlib.h

C. conio.h

D. None of the above.

All these header files are valid in C.

What will be the output of the following code snippet?

```
#include <stdio.h>
void solve() {
 printf("%d %d %d", (076), (28), (0x87));
}
int main() {
    solve();
   return 0;
}
```

A. 76 28 87

B. 076 28 0x87

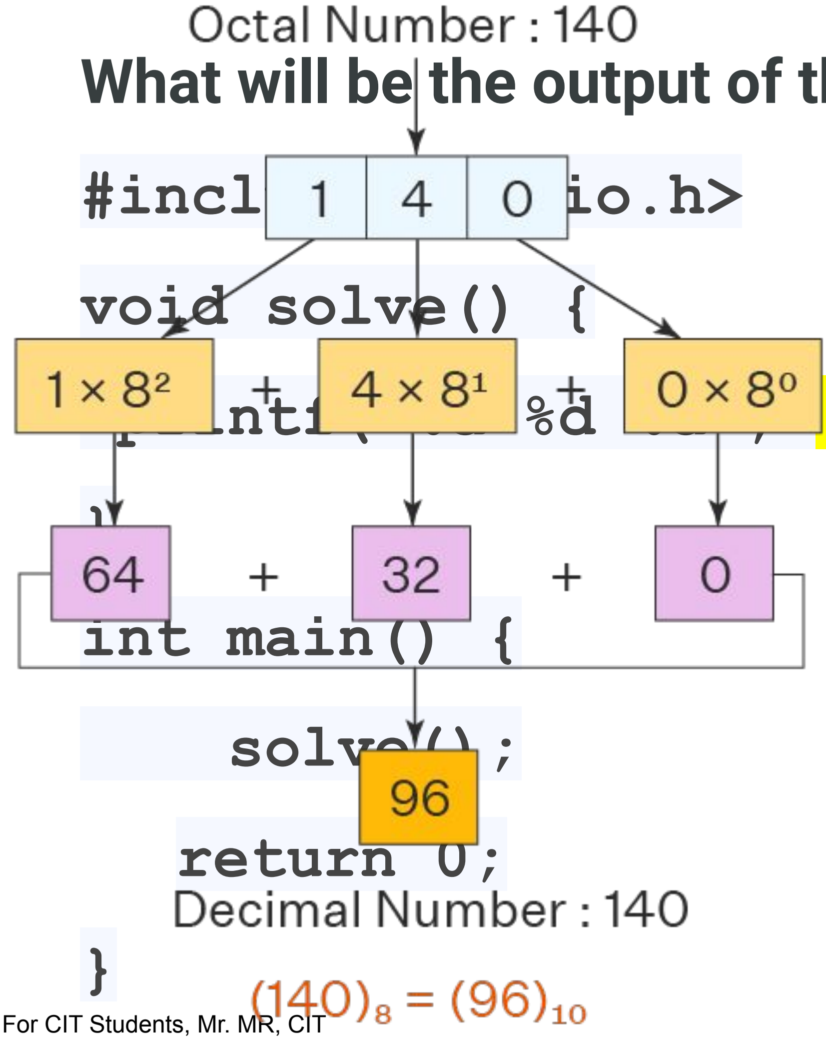
C. 62 28 135

D. 0 0 0

What will be the output of the following code snippet?

```
#include <stdio.h>
void solve() {
    printf("%d %d %d", (076), (28), (0x87));
}
int main() {
    solve();
    return 0;
}
```

Octal Decimal Hexadecimal



A. 76 28 87

B. 076 28 0x87

C. 62 28 135

D. 0 0 0

What will be the output of the following code snippet?

```
#include <stdio.h>
void solve() {
 int x = 1, y = 2;
printf(x>y? "Greater": x==y? "Equal" : "Lesser");
}

int main() {
    solve();
   return 0;
}
```

A. Greater

B. Lesser

C. Equal

D. None of the above.

What will be the output of the following code snippet?

```
#include <stdio.h>
void solve() {
 int x = 1, y = 2;
printf(x>y? "Greater": x==y? "Equal" : "Lesser");
}

int main() {
    solve();
   return 0;
}
```

Using Ternary Operator **? :**

A. Greater

B. Lesser

C. Equal

D. None of the above.

What will be the output of the following code snippet?

```
#include <stdio.h>
int main() {
  printf("%d ", 9 / 2);
  printf("%f", 9.0 / 2);
return 0;
}
```

A. 4 4.5000

B. 4 4.000

C. 4.5000 4.5000

D. 4 4

What will be the output of the following code snippet?

```
#include <stdio.h>
int main() {
  printf("%d ", 9 / 2);
  printf("%f", 9.0 / 2);
return 0;
}
```

A. 4 4.5000

B. 4 4.000

C. 4.5000 4.5000

D. 4 4

Implicitly converts to Integer & Float respectively

What will be the output of the following code snippet?

```
#include <stdio.h>
#define VAL 5
int getInput() {
    return VAL;
}

int main() {
    const int x = getInput();
    printf("%d", x);
return 0;
}
```

A. 5

B. Garbage Value

C. Compilation Error

D. 0

What will be the output of the following code snippet?

```
#include <stdio.h>
#define VAL 5
int getInput() {
    return VAL;
}

int main() {
    const int x = getInput();
    printf("%d", x);
return 0;
}
```

A. 5

B. Garbage Value

C. Compilation Error

D. 0

What will be the output of the following code snippet?

```
#include <stdio.h>
void solve(int x) {
 if(x == 0) {
  printf("%d ", x); return; }
 printf("%d ", x);
 solve(x - 1);
 printf("%d ", x);
}
int main() {
   if(x == 0) {
    printf("%d ", x);
return 0;
}
```

A. 3 2 1 0 1 2 3

B. 3 2 1 0

C. 0 1 2 3

D. None of the above

What will be the output of the following code snippet?

```
#include <stdio.h>
void solve(int x) {
 if(x == 0) {
  printf("%d ", x); return; }
 printf("%d ",
 solve(x - 1);
 printf("%d ",
}
```

```
int main() {
   if(x == 0) {
    printf("%d ", x);
return 0;
}
```

A. 3 2 1 0 1 2 3

First all the print functions before the recursive call gets executed and then all the print functions after the recursive calls get executed. Ex: smilar to STACK data structure LIFO

What will be the output of the following code snippet?

```
#include <stdio.h>
struct School {
    int age, rollNo;
};
int main() {
    struct School sc;
    sc.age = 19; sc.rollNo = 82;
    printf("%d %d", sc.age, sc.rollNo);
    return 0;    }
```

A. 19 82

B. Compilation Error

C. 82 19

D. None of the above.

What will be the output of the following code snippet?

```
#include <stdio.h>
struct School {
    int age, rollNo;
};
int main() {
    struct School sc;
    sc.age = 19; sc.rollNo = 82;
    printf("%d %d", sc.age, sc.rollNo);
    return 0;
}
```

A. 19 82

B. Compilation Error

C. 82 19

D. None of the above.

Which of the following is not true about structs in C?

A. No Data Hiding.

B. Functions are allowed inside structs.

C. Constructors are not allowed inside structs.

D. Cannot have static members in the struct body.

Which of the following is not true about structs in C?

A. No Data Hiding.

B. Functions are allowed inside structs.

C. Constructors are not allowed inside structs.

D. Cannot have static members in the struct body.

What will be the output of the following code snippet?

```
#include <stdio.h>
struct School {
    int age, rollNo;
};
void solve() {
    struct School sc;
    sc.age = 19;
    sc.rollNo = 82;
    printf("%d", (int)sizeof(sc));
}
int main() {
    solve();
      return 0;
}
```

1. 1
2. 4
3. 8
4. 16

What will be the output of the following code snippet?

```
#include <stdio.h>
struct School {
    int age, rollNo;
};
void solve() {
    struct School sc;
    sc.age = 19;
    sc.rollNo = 82;
    printf("%d", (int)sizeof(sc));
}
int main() {
    solve();
      return 0;
}
```

1. 1
2. 4
3. 8
4. 16

sum of the sizes of its individual variables.

With 2 integer types,
the size is 4 + 4 = 8bytes.

What will be the output of the following code snippet?

```
#include <stdio.h>
union School {
    int age, rollNo;
    double marks;
};
int main() {
    union School sc;
    sc.age = 19;
    sc.rollNo = 82;
    sc.marks = 19.04;
    printf("%d", (int)sizeof(sc));
   return 0;
}
```

1. 4
2. 8
3. 16
4. 12

What will be the output of the following code snippet?

```
#include <stdio.h>
union School {
    int age, rollNo;
    double marks;
};
int main() {
    union School sc;
    sc.age = 19;
    sc.rollNo = 82;
    sc.marks = 19.04;
    printf("%d", (int)sizeof(sc));
   return 0;
}
```

The size of a Union is equal to the size of the largest variable which is a part of it.
Here the variable is double which of size 8 bytes.

1. 4
2. 8
3. 16
4. 12