Coding Challenges - Troubleshoot & Debug

Scope: Practice coding knowledge for the hiring drive such as DXC

Contents

- What's expected from you?
- What type of questions are asked?
- 20 Source Codes for Practice - INCORRECT & CORRECT

What's expected from you?

The coding tests offered in job drives by many IT companies such as DXC use Automata Fix questions where you need to fix the Incorrect Source Code. The given code may have both Syntax & Logical errors. You have to figure out the mistakes in the code and execute it properly.
Follow these simple criteria to attempt to fix such codes:

1. Check syntax mistakes such as missing ; { } ( ), errors in function names.
2. Check for mistakes in loops and conditions, which may lead to infinite looping or wrong outputs.
3. Check for missing return statements.
4. Once SYNTAX errors are corrected, you start checking for LOGICAL errors. For example – the question might have asked for a greater number but you might be getting a smaller number.

What type of questions are asked?Material Compiled Ast.Prof. MR, CIT

These questions Test your Logical thinking and Problem-solving ability.
List of Common questions :

1. Arranging the data of an array in some specific order.
2. Comparing two given strings
3. Finding the second largest/smallest element of the array.
4. Finding some specific element in the given data.
5. Fixing the code for some popular searching or sorting like linear search, binary search, merge sort, radix sort, etc.

The questions will certainly vary from this list but your thorough understanding of such programs will certainly tip you to the HIRING side. So, get to work NOW!
Here are the practice programs with **both INCORRECT & CORRECTED codes** for your effective study and practice.

Problem - 1

Check for syntax error/ logical error and correct the error to get the desired output.

The function sortString modifies the input list by sorting its elements depending upon the length of the array, i.e; if the length of the array is even, then the elements are arranged in the ascending order, and if the length of the array is odd, then the elements are arranged in the descending order

The function sortString accepts two arguments – len representing the length of the string, and arr a list of characters, representing the input list respectively.

The function sortString compiles successfully but fails to get the desired results for some test cases due to logical errors. Your task is to fix the code, so that it passess all the test cases

Incorrect Source Code

```
void sortArray(int len, int *arr)
{
   int i, max, location, temp, j,k;
   if(len/2 == 0)//error in this line
   {
   for(i=0;i<len;i++)
   {
      max=arr[i];
      location = i;
      for(j=i;j<len;j++)
      if(max<arr[j])//error in this line
      {
         max=arr[j];
         location = j;

      }
      temp=arr[i];
      arr[i]=arr[location];
      arr[location]=temp;
   }
   }
   else
   {
      for(i=0;i<len;i++)
   {
      max=arr[i];
      location = i;
      for(j=i;j<len;j++) if(max>arr[j])//error in this line
```

```
        {
            max=arr[j];
            location = j;

        }
        temp=arr[i];
        arr[i]=arr[location];
        arr[location]=temp;
    }
    }
}
```

Corrected Source Code:

```
void sortArray(int len, int *arr)
{
  int i, max, location, temp, j,k;
  if(len%2 == 0)
  {
  for(i=0;i<len;i++)
  {
     max=arr[i];
     location = i;
     for(j=i;j<len;j++)
     if(max>arr[j])
     {
        max=arr[j];
        location = j;

     }
     temp=arr[i];
     arr[i]=arr[location];
     arr[location]=temp;
  }
  }
  else
  {
     for(i=0;i<len;i++)
  {
     max=arr[i];
     location = i;
     for(j=i;j<len;j++)
     if(max<arr[j])
     {
```

```
            max=arr[j];
            location = j;

        }
        temp=arr[i];
        arr[i]=arr[location];
        arr[location]=temp;
    }
    }
}
```

Problem - 2

Check for syntax error/ logical error and correct the error to get the desired output.

The function maxReplace print space separated integers representing the input list after replacing all the elements of the input list with the sum of all the element of the input list.

The function maxReplace accepts two arguments – size an integer representing the size of the input list and inputList, a list of integers representing the input list respectively.

The function maxReplace compiles unsuccessfully due to compilation errors. Your task is to fix the code so that it passes all the test cases.

Incorrect Source Code

```
void maxReplace(int size, int *inputList)
{
    int i,sum=0;
    for(i=0;i<size;i++)
    {
        sum += inputList[i];
    }
        for(i=0;i<size;i++)
        {
            sum = inputList[i];//error in this line
        }
    for(i=0;i<size;i++)
    {
        printf("%d ",inputList[i]);
    }
}
```

Corrected Source Code:

```
void maxReplace(int size, int *inputList)
{
   int i,sum=0;
   for(i=0;i<size;i++)
   {
       sum += inputList[i];
   }
      for(i=0;i<size;i++)
      {
         inputList[i]=sum;
      }
   for(i=0;i<size;i++)
   {
      printf("%d ",inputList[i]);
   }
}
```

Problem - 3

Check for syntax error/ logical error and correct the error to get the desired output.

The function replaceElements is modifying the input list in such a way – if the sum of all the elements of the input list is odd, then all the elements of the input list are supposed to be replaced by 1s, and in case if the sum of all the elements of the input list is even, then the elements should be replaced by 0s.

For example, given the input list [1,2,3,4,5], the function will modify the input list like [1, 1, 1, 1, 1]
The function replaceElements accepts two arguments – size an integer representing the size of the given input list and arr, a list of integers representing the input list.

The function replaceElements compiles successfully but fails to get the desired result for some test cases due to incorrect implementation of the function. Your task is to fix the code so that it passes all the test cases

Incorrect Source Code

```
void replaceElements(int size, int *arr)
{
   int i,j;
   int sum=0;
   for (i=0;i<size;i++)
```

```
    {
        sum+=arr[i];
    }
    if(size % 2 == 0)//error in this line
    {
        i=0;
        while(i<size)
        {
            arr[i] = 0;
            i += 1;

        }
    }
    else
    {
        j=1;
        while(j<size)
        {
            arr[j]=1;
            j+=1;
        }
    }
```

Corrected Source Code

```
void replaceElements(int size, int *arr)
{
   int i,j;
   int sum=0;
   for (i=0;i<size;i++)
   {
       sum+=arr[i];
   }
   if(sum % 2 == 0)
   {
       i=0;
       while(i<size)
       {
           arr[i] = 0;
           i += 1;

       }
   }
   else
```

```
{
   j=1;
   while(j<size)
   {
      arr[j]=1;
      j+=1;
   }
}
```

Problem - 4

Ram is a 6 years old boy who loves to play with numeric lego. One day ram's mom created a number using those lego and asked ram to tell the number of elements available between two specific numbers 'alpha1' and 'alpha2'. After 15 years when ram started learning C, he now wants to write a C code to find the number of elements lies between ranges alpha1 and alpha2. If the number is arr and the starting and ending points are alpha1 and alpha2, find the numbers of elements lies in the range

Input:
Three space-separated integers
1. First is the length of arr.
2. Second is the starting point as alpha1
3. The third is the endpoint as alpha2

Output:
Indexes of elements lie between this range.

Example
Input
9 2 6
1 2 3 4 5 6 7 8 9
Output
1 2 3 4
Find the error in the given code

Incorrect Source Code

```
#include

int main()

{

   int starting_point, ending_point, arr[20], length, j;
```

```
    scanf("%d %d %d",&length,&starting_point,&ending_point);
    for(j=0; j<=length; j++)
    {
        scanf("%d",&arr[j]);
    }
    for(j=0;j<length;j++)
    {
        if(arr[j]>=starting_point && arr[j]<=ending_point)
        {
            printf("%d ",j);
        }
    }
    return 0;
}
```

Correct Source Code

```
#include<stdio.h>
int main()
{
    int starting_point, ending_point, arr[20], length, j;
    scanf("%d %d %d",&length,&starting_point,&ending_point);
    for(j=0; j<length; j++)
    {
```

```
        scanf("%d",&arr[j]);
    }
    for(j=0;j<length;j++)
    {
        if(arr[j]>=starting_point && arr[j]<ending_point)
        {
            printf("%d ",j);
        }
    }
    return 0;
}
```

Problem - 5

Find the security key to access the bank account in from the encrypted message. The key in the message is the first repeating number from the given message of numbers.

Input format:
Single Integer value
Output format:
The first repeating number from the message
Example:
Input:
123456654321
Output:
1
Find the error in the given C programming code:

Incorrect Source Code

```
#include

#include
```

```
int main()
{
    char arr[20];
    int x, y, len, flag=0;
    scanf("%s",arr);
    len = strlen(arr);
    for(x=0;x<len;x++)
    {
        for(y=x+1;y<len;y++)
        {
            if(arr[x]==arr[y] && flag == 0)
            {
                printf("%c",arr[y])
                flag = 1;
                break;
            }
        }
    }
    return 0;
}
```

Correct Source Code

```
#include <stdio.h>

#include <string.h>

int main()

{

    char arr[20];

    int x, y, len, flag=0;

    scanf("%s",arr);

    len = strlen(arr);

    for(x=0;x<len;x++)

    {

        for(y=x+1;y<len;y++)

        {

            if(arr[x]==arr[y] && flag == 0)

            {

                printf("%c",arr[y]);

                flag = 1;

                break;

            }

        }

    }

    return 0;
```

}

Problem - 6

From the given set of the array, integer numbers in an array Write the program to add the numbers the same as the given number. The program takes 3 space-separated values, First is the length as 'len', Second arr[]ar as 'arr', Third the given number as 'value'.

Input format:
Three space-separated inputs

1. Length
2. Input number
3. Element

Output format:
Sum of elements the same as the value
Example:
Input:
8 12342562 2
Output:
6

Incorrect Source Code

```
int sumOfValue(int len, int* arr, int value)

{

   int sum = 0;

   for(int i =0 ; i < len -1 ;   )

   {

      if(arr[i]==value)

         sum += value;

   }

   return sum;

}
```

Correct Source Code

```
int sumOfValue(int len, int* arr, int value)

{

   int sum = 0;

   for(int i =0 ; i < len ; i++)

   {

      if(arr[i]==value) sum += value;

   }

   return sum;

}
```

Problem - 7

For the generation of energy at the atomic center the energy is used to start the plant is in negative sign and the energy generated after the plant started is represented with a positive sign. Our task is to calculate the total number of energy that is generated throughout the day. Below is the code in C programming but it is not working well find the error in the code

Input format:
Two inputs One for the number of times energy was recorded as 'total_reading' and second the reading of the energy used or produced 'arr'

Output format:
Sum of the total energy produced
Example:
Input:
8
1 4 -5 6 7 -4 4 -1
Output:
12

Incorrect Source Code

```
#include <stdio.h>

int main()

{

int total_reading, i, arr[20];

scanf("%d",&total_reading);

for(i=0; i<=total_reading; i++)

{

    scanf("%d",&arr[i]);

}

int sum = 0;

for(i = 0;i<total_reading;i++){

    sum+=arr[i];

}

printf("%c",sum);

return 0;

}
```

Correct Source Code

```
#include <stdio.h>

int main()

{

int total_reading, i, arr[20];

scanf("%d",&total_reading);
```

```
for(i=0; i<total_reading; i++)

{

    scanf("%d",&arr[i]);

}

int sum = 0;

for(i = 0;i<total_reading;i++){

    sum+=arr[i];

}

printf("%d",sum);

return 0;

}
```

Problem - 8

Print the number at the given index of the series 1 1 2 3 5… and so on. Consider the index as the number and print the value

Input format:
Single input of the index of the number as ‘number’

Output format:
The number at the given index of the series
Example:
Input:
10
Output:
55
below is the code for this problem find the problem in the given code

Incorrect Source Code

```
#include <stdio.h>

int main()
```

```
{
    int val1 = 0, val2 = 1, val3 = 1;
    int number, i;
    scanf("%d"&number);
    for(i=2; i<=number; i++)
    {
        val3 = val1 +val2;
        val1 = val2;
        val2 = val3;
          }
 printf("%d ",&val3);
    return 0;
}
```

Correct Source Code

```
#include <stdio.h>
int main()
{
    int val1 = 0, val2 = 1, val3 = 1;
    int number, i;
    scanf("%d",&number);
    for(i=2; i<=number; i++)
    {
```

```
        val3 = val1 +val2;

        val1 = val2;

        val2 = val3;


    }

printf("%d ",val3);

    return 0;

}
```

Problem - 9

Function/method employeeID accepts four arguments-len , an integer representing the length of input list. arr, start , end of range and a list of integers It returns an integer representing the sum of all id's of the employees in that range for example

len = 6,start=30,end=50, arr = [29 38 12 48 39 55]
function /method will return 8 i.e 1+3+4. Function/method compiles successfully but fails to return the desired result for some/all cases due to incorrect implementation. Your task is to debug the code so that it passes all test cases.

Example
Input
6 30 50
29 38 12 48 39 55
Output
8
Explanation :
There are 3 employees with id 1, 3, 4 whose distance from the office lies within the given range sum of the id's is 8.

Incorrect Source Code

```
int employeeID (int len, int start, int end, int *arr)
{

   for (i = 0; i < len; i++)
```

```
  {
     if (arr[i] > start || arr[i] < end)
         {
         flag = 1;
     }
     else
         {
         flag = -1;
     }

     if (flag == -1)
         {
         s = s + i;
     }
  }
  return s;
}
```

Correct Source Code

```
int employeeID (int len, int start, int end, int *arr)
{

  for (i = 0; i < len; i++)
  {

     if (arr[i] > start && arr[i] < end)
         {
         flag = 1;
     }
     else
         {
         flag = -1;
     }

     if (flag == 1)
         {
         s = s + i;
     }
  }
  return s;
}
```

Problem - 10

Function/method secretKey accepts a single argument- arr , a string. It returns an integer representing the count of all characters in it without duplicates(length of string after removing duplicates)

arr=5435436789
function /method will return 7. Function/method compiles successfully but fails to return the desired result for some/all cases due to incorrect implementation. Your task is to debug the code so that it passes all test cases.

Example
Input
5435436789
Output
7
Explanation
The repeated digits in the data are 5,4 and 3. So, the security key is 7 which is the length of the string after removing duplicates.

Incorrect Source Code

```
int securityKey (char *a)
{

  int i, j, len, count = 0;
  len = strlen (a);
  for (i = 0; i < len; i++)
  {
     int flag = 1;
     for (j = 0; j < len; j++)
         {
        if (a[i] == a[j])
            {
           break;
        }
     }
     if (flag == 1)
         {
        count++;
     }
  }
  return count;
}
```

Correct Source Code

```
int securityKey (char *a)
{
  int i, j, len, count = 0;
  len = strlen (a);

  for (i = 0; i < len; i++)
  {
     int flag = 1;
     for (j = i + 1; j < len; j++)
         {
         if (a[i] == a[j])
             {
             flag = 0;
             break;
         }
     }
     if (flag == 1)
         {
         count++;
     }
  }
  return count;
}
```


Problem - 11

Function/method energyCal accepts two arguments-len , an integer representing the length of input list. energy, a list of integers. It returns an integer representing the sum of elements whose product is maximum for example
len = 7, energy= [9 -3 8 -6 -7 8 10]
function /method will return 19 function/method compiles successfully but fails to return the desired result for some/all cases due to incorrect implementation. Your task is to debug the code so that it passes all test cases.

Example
Input
7
9 -3 8 -6 -7 8 10
Output
19
Explanation
The maximum product of the energies is 90 i.e 9*10

So the sum of energy of chemicals is 19.

Incorrect Source Code

```
int energyCal (int numOfChem, int *energy)
{

  int numOfChem, energy[100], i, j, result, temp;
  for (i = 0; i < numOfChem; i++)
  {
     for (j = i + 1; j < numOfChem + 1; j++)
         {
        if (energy[i] <= energy[j])
            {
            temp = energy[i];
            energy[i] = energy[j];
            energy[j] = temp;
        }
     }
  }
  result = energy[numOfChem - 1] + energy[numOfChem - 2];
  return result;
}
```

Correct Source Code

```
int energyCal (int numOfChem, int *energy)
{

  int numOfChem, energy[100], i, j, result, temp;
  for (i = 0; i < numOfChem; i++)
  {
     for (j = i + 1; j < numOfChem; j++)
         {
        if (energy[i] > energy[j])
            {
            temp = energy[i];
            energy[i] = energy[j];
            energy[j] = temp;
        }
     }
  }
  result = energy[numOfChem - 1] + energy[numOfChem - 2];
  return result;
```

}

Problem - 12

Function/method nthFib accepts single argument-num , an integer representing number. for example num=9
function /method will return 34 function/method compiles successfully but fails to return the desired result for some/all cases due to incorrect implementation. Your task is to debug the code so that it passes all test cases.

Example
Input
8
Output
21
Explanation:
The sequence generated by the system will be 1, 1, 2, 3, 5, 8, 13, 21. The 8th number generated from this series will be 21

Incorrect Source Code

```
int nthFib ()
{
  int a = 0;
  int b = 1;
  int c = 1;
  int num, i;

  for (i = 0; i <= num; i++)
  {
     c = a + b;
     a = b;
     b = c;
  }
  return a + b;
}
```

Correct Source Code

```
int nthFib ()
{
  int a = 0;
  int b = 1;
  int c = 1;
  int num, i;

  for (i = 2; i <= num; i++)
  {
     c = a + b;
     a = b;
     b = c;
  }
  return c;
}
```

Problem - 13

The function studentSort(int *arr, int len) accepts an integer array arr(which is heights of students) of length len as an input and performs an in place sort operations on it. The function is expected to return the input array sorted in descending order of their heights, but instead, it returns the array sorted in ascending order due to a bug in the code.

Your task is to debug the program to pass all test cases.

Incorrect Source Code

```
int * studentSort (int arr[], int len)
{

  int small, pos, i, j, temp;
  for (i = 0; i < len; i++)
  {
     for (j = 0; j < len; j++)
         {
        temp = 0;
        if (arr[i] <= arr[j])
            {
           temp = arr[j];
           arr[i] = arr[j];
           arr[j] = temp;
        }
```

```
    }
  }
  return arr;
}
```

Correct Source Code

```
int * studentSort (int arr[], int len)
{

  int small, pos, i, j, temp;
  for (i = 0; i < len; i++)
  {
     for (j = 0; j < len; j++)
         {
        temp = 0;
        if (arr[i] > arr[j])
            {
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
     }
  }
  return arr;
}
```


Problem - 14

The function patternPrint(int num) prints even or odd numbers based on the values of the input arguments num(num >= 0).

If the input number num is even, the function is expected to print the even whole numbers upto num and in case it is odd, is expected to print the odd numbers upto num(inclusively).

For example given input 6, the function prints the string “0 2 4 6”(without quotes).
The function compiles successfully but fails to print the desired result due to logical errors.

Your task is to debug the program to pass all the test cases.

Test Cases:
Test Case 1:
Input:
23

Expected Return Value:
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45

Test Case 2:
Input:
14
Expected Return Value:
0 2 4 6 8 10 12 14 16 18 20 22 24 26

Incorrect Source Code

```
void printPattern (int num)
{

  int i, print = 0;
  if (num % 2 == 0)
  {
     print = 0;
     for (i = 0; i <= num; i++)
         {
         printf ("%d ", print);
     }
  }
  else
  {
     print = 1;
     for (i = 0; i <= num; i++)
         {
         printf ("%d ", print);
     }
  }
}
```

Correct Source Code

```
void printPattern (int num)
{
  int i, print = 0;
  if (num % 2 == 0)
  {
     print = 0;
     for (i = 0; print <= num; i++)
         {
         printf ("%d ", print);
```

```
        print += 2;
      }
   }
   else
   {
      print = 1;
      for (i = 0; print <= num; i++)
          {
         printf ("%d ", print);
         print += 2;
      }
   }
}
```

Problem - 15

The function mulTable(int num) is supposed to print the first twenty multiples of the multiplication table of the input number num

.

The function compiles fine but fails to return the desired result for some cases.

Your task is to fix the program so that it passes all the test cases.

Test cases:

TestCase 1:
Input:
6
Expected return value:
6 12 18 24 30 36 42 48 57 60 66 72 78 84 90 96 102 108 114 120

TestCase 2:
Input:
0
Expected return value:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Incorrect Source Code

```
void printTable (int num)
{
   int value = 0;
   for (int i = 0; i < 10; i++)
   {
```

```
      value = num * num;
      printf ("%d ", value);
   }
}
```

Correct Source Code

```
void printTable (int num)
{
   int value = 0;
   for (int i = 1; i <= 20; i++)
   {
      value = num * i;
      printf ("%d ", value);
   }
}
```

Problem - 16

The function binarySearch(int* arr,int len, int target) performs the binary search algorithm to look for an element target in the input array arr of length len . In case it is found the function returns the index of target in arr and returns a-1 if not found.

The function works seemingly well but goes into an infinite loop for some test cases.

Your task is to fix the program so that it passes all the test cases.

Note: If there is a Time limit exceeded error, it can be due to an infinite loop.

Test Cases:

TestCase 1:
Input:
[0 , 2 , 3 , 4 , 5, 6 , 7 , 8] , 8 , 4
Expected Return Value:
3

TestCase 2:
Input:
[2 , 3],2,3
Expected Return Value:
1

Incorrect Source Code

```
int binarySearch (int *ar, int len, int target)
{
   int lo = 0, hi = len;
   while (lo <= hi)
   {
      int mid = (lo + hi);
      if (ar[mid] == target)
          {
         return mid;
      }
      else
          {
         if (ar[mid] < target)
             {
            lo = lo - 1;
         }
          else
             {
            hi = mid - 1;
         }
      }
   }
   return -1;
}
```

Correct Source Code

```
int binarySearch (int *ar, int len, int target)
{

   int lo = 0, hi = len;
   while (lo <= hi)
   {
      int mid = (lo + hi) / 2;
      if (ar[mid] == target)
          {
         return mid;
      }
      else
          {
         if (ar[mid] < target)
             {
```

```
            lo = mid;
          }
           else
              {
              hi = mid - 1;
          }
       }
   }
   return -1;
}
```

Problem - 17

The function/method sameOddElementCount returns an integer representing the number of elements of the input list which are odd numbers and equal to the elements to its left. For example, if the input list is [1,3,3,4,5,5,9,9,10,11] then the function/method should return the output '3' as it has three similar groups i.e. {3,3}, {5,5}, {9,9}.

The function/method sameOddElementCount accepts two arguments – size, and integer representing the size of the input list and inputList, a list of integers representing the input list.
The function/Method compiles successfully but fails to return the desired result for some test cases due to incorrect implementation of the function/method sameOddElementCount. Your task is to fix the code so that it passes all the test cases.

Test Case 1:
Input :
11
[1, 5, 5, 2, 2, 7,7, 8, 6 ,6, 9, 10]
Expected Return Value
2

Test Case 1:
Input :
5
[13, 12, 12, 13, 14]
Expected Return Value
0

Incorrect Source Code

```
int sameOddElementCount (int size, int *inputList)
{

   int i, count = 0;
```

```
  for (i = 1; i < size - 1; i++)
  {
     if ((inputList[i] % 2 == 0) && (inputList[i] == inputList[i--]))
        count++;
  }
  return count;
}
```

Correct Source Code

```
int sameOddElementCount (int size, int *inputList)
{
  int i, count = 0;
  for (i = 1; i < size; i++)
  {
     if ((inputList[i] % 2 == 1) && (inputList[i] == inputList[i - 1]))
        count++;
  }
  return count;
}
```

Problem - 18

The function/method allExponent returns a real number representing the result of exponentiation of base raised to power exponent for all input values.

The function/method allExponent accepts two arguments – baseValue, an integer representing the base and exponentValue, and integer representing the exponent.

The incomplete code in the function/method allExponent works only for negative values of the exponent. You must complete the code and make it work for positive values of exponent as well. Another function negativeExponent uses an efficient way for exponentiation but accepts only negative exponent values. You are supposed to use this function/method to complete the code in allExponent function/method.

Incorrect Source Code

```
float allExponent (int baseValue, int exponentValue)
{
  float res = 1;
  if (exponentValue < 0)
  {
     res = (float) negativeExponent (baseValue, exponentValue);
  }
```

```
  else
  {
        // write your code for negative value
  }
  return res;
}
```

Correct Source Code

```
float allExponent (int baseValue, int exponentValue)
{
  float res = 1;
  if (exponentValue < 0)
  {
     res = (float) negativeExponent (baseValue, exponentValue);
     // is to find exponents of negative numbers
  }
  else
  {
     res = (float) 1 / (negativeExponent (baseValue, exponentValue));
        // find negative exponents and then find reciprocals of it to get positive exponent
  }
  return res;
}
```

Problem 19

You are required to fix all errors in the given code:The function/method multiplyNumber The function/method returns an integer representing the multiplicative product of the maximum two of three input numbers.The function/method multiplyNumber accepts three integers-numA,numB,and numC representing the input numbers.

multiplyNumber complies unsuccessfully due to syntactical error. Your task is to debug the code so that it passes all the test cases.

Testcase 1:
Input

5,7,4

Expected Return Value
35

Testcase 2:
Input

11,12,13

Expected Return Value
156

Incorrect Source Code:

```
//You can print the values to stdout for dubugging.
int mutipleNumber(int numA, int numB, int numC)
{
int result , min, max,mid;
max=(numA>numB)?((numA>numC)? numA:numC): ((numB>numC)?(numB
min=(numA<numB)?((numA<numC)? numA:numC): ((numB<numC)?(numB
mid=(numA+numB+numC)-(min+max);
result = (max*mid);
return result;
}
```

Correct Source Code for the exam:

```
int mutiplyNumber(int numA, int numB, int numC)

{
int result,min,max,mid;
max=(numA>numB)?((numA>numC)? numA:numC): ((numB>numC)?numB:numC);
min=(numA<numB)?((numA<numC)? numA:numC): ((numB<numC)?numB:numC);
mid=(numA+numB+numC)-(min+max);
printf("Min: %d Mid: %d Max: %d ",min,mid,max);
result = (max*mid);
return result;
}
```

Correct FULL Source Code for compiling on your PC:

```
#include <stdio.h>
int mutiplyNumber(int numA, int numB, int numC);
int main()
   {
    printf("\nMax * Mid : %d",mutiplyNumber(10,20,30));
```

```
        return 0;
    }

//You can print the values to stdout for dubugging.

int mutiplyNumber(int numA, int numB, int numC)

{
int result,min,max,mid;
max=(numA>numB)?((numA>numC)? numA:numC): ((numB>numC)?numB:numC);
min=(numA<numB)?((numA<numC)? numA:numC): ((numB<numC)?numB:numC);
mid=(numA+numB+numC)-(min+max);
printf("Min: %d Mid: %d Max: %d ",min,mid,max);
result = (max*mid);
return result;
}
```

Problem 20

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.

Incorrect Source Code:

```
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. The modified linked list after the function call should be 6 <--> 5 <--> 4 <--> 3 <--> 2 <-->1

Correct Source Code:

```
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;
}
```
