```
You are required to fix all the logical error in the given code. You can click on Compile & Run anytime to check the compilation/execution status of the program. You can use printf() to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the main() function as it is not required.

Code Approach: For this question, you will need to complete the code as in given implementation. We do not expect you to modify the approach.

The function/method patternPrint accepts an argument num, an integer.

The function/method patternPrint prints num lines in the following pattern.

For example, num = 4, the pattern should be:
```

The function/method *patternPrint* compiles successfully but fails to print the desired result for

111

some test cases due to incorrect implementation of the function/method. Your task is to fix the code so that it passes all the test cases.

```
You can print the values to stdout for debugging
    void patternPrint(int num)
        int print=1,i,i;
        for(i=0;i<num;i++)
 6 -
            for(j=0;j<=i;j++);
                 printf("%d ",print);
10
11
            printf("\n");
12
13
14
15
```

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *printf* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the *main()* function as it is not required.

Code Approach: For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

Lisa always forgets her birthday which is on the 5th of July. So, develop a function/method which will be helpful to remember her birthday.

The function/method **checkBirthDay** return an integer '1' if it is her birthday else returns 0. The function/method **checkBirthDay** accepts two arguments - *month*, a string representing the month of her birthday and *day*, an integer representing the date of her birthday.

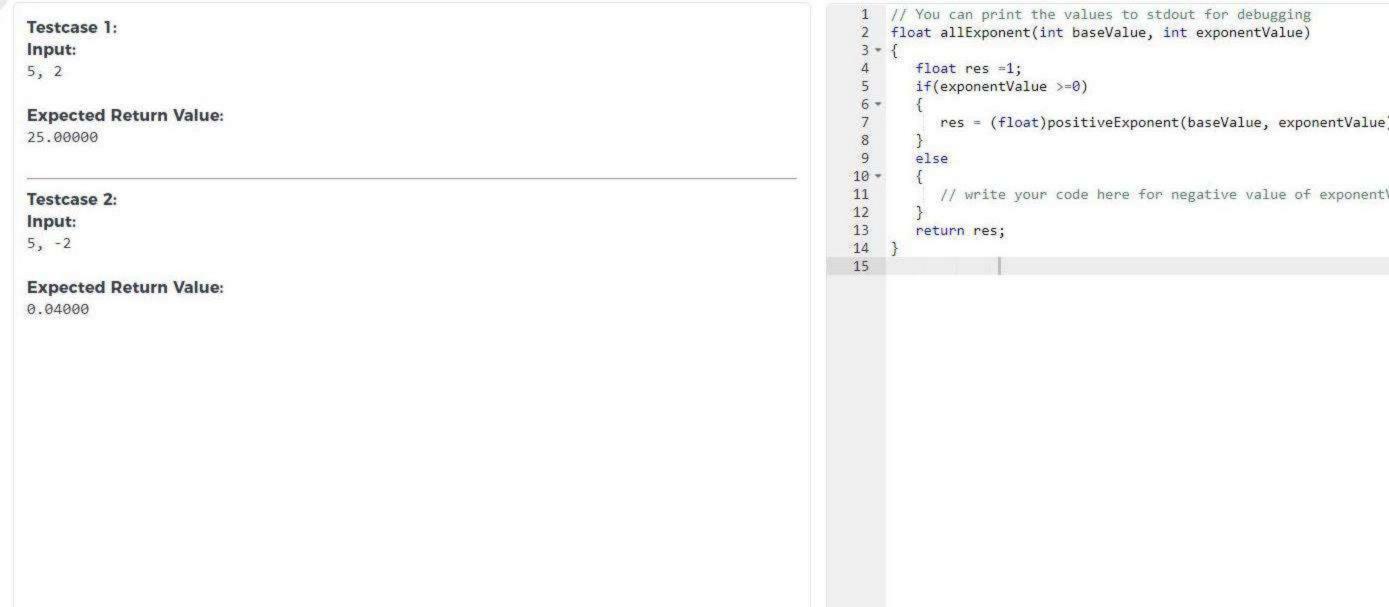
result for some test cases. Your task is to fix the code so that it passes all the test cases,

The function/method **checkBirthDay** compiles successfully but fails to return the desired

```
You are required to complete the given code. You can click on Compile & Run anytime
to check the compilation/execution status of the program. You can use printf() to debug
                                                                                                  3 -
your code. The submitted code should be logically/syntactically correct and pass
all testcases. Do not write the main() function as it is not required.
Code Approach: For this question, you will need to complete the code as in given
implementation. We do not expect you to modify the approach.
The function/method allExponent returns a real number representing the result of
                                                                                                 10 -
exponentiation of base raised to power exponent for all input values. The function/method
                                                                                                 11
allExponent accepts two arguments - baseValue, an integer representing the base and
                                                                                                 12
exponent Value, an integer representing the exponent.
                                                                                                 13
                                                                                                 14
                                                                                                 15
The incomplete code in the function/method allExponent works only for positive values of the
exponent. You must complete the code and make it work for negative values of exponent as
well.
Another function/method positiveExponent uses an efficient way for exponentiation but
accepts only positive exponent values. You are supposed to use this function/method to
complete the code in allExponent function/method.
Helper Description
The following function is used to represent a positive Exponent and is already implemented in
the default code (Do not write this definition again in your code):
int positiveExponent(int baseValue, int exponentValue)
    /*It calculates the Exponent for the positive value of exponentValue
      This can be called as -
      int res = (float)positiveExponent(baseValue, exponentValue); */
```

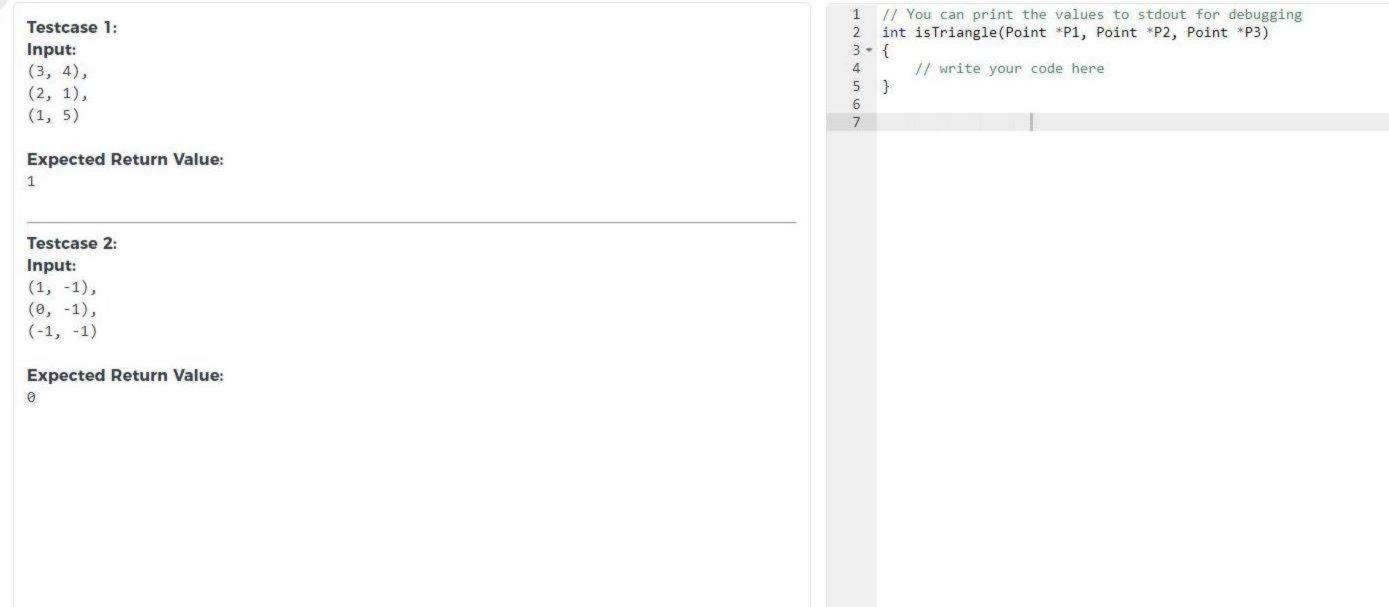
```
float res =1:
if(exponentValue >=0)
   res = (float)positiveExponent(baseValue, exponentValue
else
   // write your code here for negative value of exponent
return res;
```

float allExponent(int baseValue, int exponentValue)



```
You are required to complete the given code. You can click on Compile &
Run anytime to check the compilation/execution status of the program. You can
                                                                                                 3 - {
use printf to debug your code. The submitted code should be logically/syntactically
correct and pass all testcases. Do not write the main() function as it is not required.
Code Approach: For this question, you will need to complete the code as in given
implementation. We do not expect you to modify the approach.
You are given a predefined structure Point and also a collection of related
functions/methods that can be used to perform some basic operations on the structure.
You must implement the function/method isTriangle which accepts three points P1, P2, P3
as inputs and checks whether the given three points form a triangle.
If they form a triangle, the function/method returns an integer 1. Otherwise, it returns an
integer 0.
Helper Description
The following structure is used to represent point and is already implemented in the default
code (Do not write these definitions again in your code):
struct point;
typedef struct point
 int X;
 int Y;
}Point;
double Point calculateDistance(Point *point1, Point *point2)
```

```
// You can print the values to stdout for debugging
int isTriangle(Point *P1, Point *P2, Point *P3)
   // write your code here
```



You are required to fix all syntactical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *printf()* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the *main()* function as it is not required.

Code Approach: For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

The function/method **matrixSum** returns an integer representing the sum of elements of the input matrix. The function/method **matrixSum** accepts three arguments - rows, an integer representing the number of rows of the input matrix, columns, an integer representing the number of columns of the input matrix and matrix, a two-dimensional array representing the input matrix.

to debug the program so that it passes all test cases.

The function/method matrixSum compiles unsuccessfully due to syntactical error. Your task is

```
1  // You can print the values to stdout for debugging
2  int matrixSum(int rows, int columns, int **matrix)
3 * {
4    int i, j, sum=0;
5    for(i=0;i<rows;i++)
6 * {
7     for(j=0;j<columns;j++)
8     sum + = matrix(i)(j);
9    }
10    return sum;
11 }</pre>
```

```
1 // You can print the values to stdout for debugging
Testcase 1:
                                                                                                        int matrixSum(int rows, int columns, int **matrix)
                                                                                                     3 - {
4
5
6 -
Input:
                                                                                                            int i, j, sum=0;
3, 3,
                                                                                                            for(i=0;i<rows;i++)
[[3, 2, 1],
 [4, 6, 5],
                                                                                                    7
8
9
                                                                                                              for(j=0;j<columns;j++)</pre>
 [7, 8, 9]]
                                                                                                                sum + = matrix(i)(j);
                                                                                                    10
                                                                                                            return sum;
Expected Return Value:
                                                                                                   11 }
45
Testcase 2:
Input:
3, 3,
[[3, 12, 10],
 [14, 61, 51],
 [27, 84, 95]]
Expected Return Value:
357
```

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *printf()* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases.

Code Approach: For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library

methods.

The function/method **selectionSortArray** performs an in-place selection sort on the given input list which will be sorted in ascending order.

The function/method **selectionSortArray** accepts two arguments - *len*, an integer representing the length of the input list and *arr*, a list of integers representing the input list, respectively.

The function/method **selectionSortArray** compiles successfully but fails to get the desired result for some test cases due to logical errors. Your task is to fix the code so that it passes all the test cases.

Note:

In this particular implementation of selection sort, the smallest element in the list is swapped with the element at first index, the next smallest element is swapped with the element at the next index and so on.

```
You can print the values to stdout for debugging
    void selectionSortArray(int len, int* arr)
3 - {
        int x=0, y=0;
        for(x=0; x<len; x++){
           int index of min = x;
           for(y=x; y<len; y++){
              if(arr[index of min]>arr[x]){
                 index of min = y;
10
11
12
           int temp = arr[x];
13
           arr[x] = arr[index of min];
14
           arr[index of min] = temp;
15
16
17
```

```
// You can print the values to stdout for debugging
Testcase 1:
                                                                                                     void selectionSortArray(int len, int* arr)
                                                                                                  3 - {
Input:
                                                                                                         int x=0, y=0;
10, [3, 6, 4, 1, 7, 9, 1, 3, 12, 15]
                                                                                                  5 =
                                                                                                         for(x=0; x<len; x++){
                                                                                                            int index_of_min = x;
Expected Return Value:
                                                                                                            for(y=x; y<len-1; y++){
1 1 3 3 4 6 7 9 12 15
                                                                                                               if(arr[index_of_min]>arr[x]){
                                                                                                  8 +
                                                                                                                  index of min = y;
                                                                                                 10
                                                                                                 11
Testcase 2:
                                                                                                 12
                                                                                                            int temp = arr[x];
Input:
                                                                                                 13
                                                                                                            arr[x] = arr[index_of_min];
9, [3, 3, 3, 3, 3, 3, 3, 3]
                                                                                                 14
                                                                                                            arr[index_of_min] = temp;
                                                                                                 15
                                                                                                 16
Expected Return Value:
                                                                                                 17
3 3 3 3 3 3 3 3 3
```

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *printf()* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases.

Code Approach: For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library

methods.

The function/method *descendingSortArray* performs an in-place sort on the given input list which will be sorted in descending order.

The function/method *descendingSortArray* accepts two arguments - *len*, an integer

representing the length of the input list and *arr*, a list of integers representing the input list, respectively.

The function/method **descendingSortArray** compiles successfully but fails to get the desired result for some test cases due to logical errors. Your task is to fix the code so that it passes all the test cases.

```
You can print the values to stdout for debugging
    void descendingSortArray(int len, int* arr)
3 - {
        int small, pos, i, j, temp;
        for(i=0; i<=len-1;i++){
           for(j=i; j<len;j++){
             temp = 0;
             if(arr[i]>arr[j]){
               temp=arr[i];
10
               arr[i]=arr[j];
               arr[j]=temp;
11
12
13
14
15
16
```

Run anytime to check the compilation/execution status of the program, You can use printf() to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the main() function as it is not required.

Code Approach: For this question, you will need to complete the code as in given implementation. We do not expect you to modify the approach.

The function/method sameElementCount returns an integer representing the number of elements of the input list which are even numbers and equal to the element to its right. For example, if the input list is [4 4 418 411 2 2] then the function/method should return the output '3' as it has three similar groups i.e, (4, 4), (4, 4), (2, 2).

The function/method sameElementCount accepts two arguments - size, an integer representing the size of the input list and inputList, a list of integers representing the input list.

You are required to fix all the logical error in the given code. You can click on Compile &

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

Note: In a list, an element at index i is considered to be on the left of index i+1 and to the right of

index i-1. The last element of the input list does not have any element next to it which makes it incapable to satisfy the second condition and hence should not be counted.

```
// You can print the values to stdout for debugging
int sameElementCount(int size, int *inputList)
    int i,count =0;
    for(i=0;i<size-1;i++)
       if((inputList[i]%2==0)&&(inputList[i]==inputList[i++]
          count++;
    return count;
```

10

11

12

13



You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *printf()* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases.

Code Approach: For this question, you will need to correct the given implementation.

We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method **countOccurrence** return an integer representing the count of occurrences of given value in the input list.

The function/method **countOccurrence** accepts three arguments - *len*, an integer representing the size of the input list, *value*, an integer representing the given value and *arr*, a list of integers, representing the input list.

result for some test cases due to logical errors. Your task is to fix the code so that it passes all the test cases.

The function/method *countOccurrence* compiles successfully but fails to return the desired

```
1  // You can print the values to stdout for debugging
2  int countOccurrence( int len, int value, int *arr)
3 * {
4    int i=0, count = 0;
5 * while(i<len){
6    if(arr[i]==value)
7    count += 1;
8    }
9    return count;
10 }</pre>
```

```
// You can print the values to stdout for debugging
Testcase 1:
                                                                                                     int countOccurrence( int len, int value, int *arr)
Input:
                                                                                                 3 - {
                                                                                                       int i=0, count = 0;
7, 3, [2, 3, 4, 3, 5, 6, 7]
                                                                                                       while(i<len){
                                                                                                          if(arr[i]==value)
Expected Return Value:
                                                                                                             count += 1;
                                                                                                 8
                                                                                                 9
                                                                                                       return count;
                                                                                                10
                                                                                                11
Testcase 2:
Input:
1, 2, [9]
Expected Return Value:
```

```
You are required to complete the given code. You can click on Compile & Run anytime
to check the compilation/execution status of the program. You can use printf() to debug
your code. The submitted code should be logically/syntactically correct and pass
all testcases. Do not write the main() function as it is not required.
Code Approach: For this question, you will need to complete the code as in given
implementation. We do not expect you to modify the approach.
The function/method median accepts two arguments - size and inputList, an integer
representing the length of a list and a list of integers, respectively.
The function/method median is supposed to calculate and return an integer representing the
median of elements in the input list. However, the function/method median works only for
odd-length lists because of incomplete code.
You must complete the code to make it work for even-length lists as well. A couple of other
functions/methods are available, which you are supposed to use inside the function/method
median to complete the code.
Helper Description
The following function is used to represent a quick_select and is already implemented in the
default code (Do not write this definition again in your code):
int quick select(int* inputList, int start index, int end index, int median order)
   /*It calculate the median value
   This can be called as -
   quick select(inputList, start index, end index, median order)
   where median order is the half length of the inputList
```

```
// You can print the values to stdout for debugging
    float median(int size, int * inputList)
        int start index = 0;
        int end index = size-1;
        float res = -1;
        if(size%2!=0) // odd size inputList
           int median order = ((size+1)/2);
           res = (float)quick select(inputList, start index, end
        else // even size inputList
13 -
            // Write code here
        return res;
```

10

11 12

14

15

16

17

18

19

```
Testcase 1:
                                                                                                       float median(int size, int * inputList)
Input:
                                                                                                           int start index = 0;
5,
                                                                                                           int end index = size-1;
[2, 40, 23, 52, 37]
                                                                                                           float res = -1;
                                                                                                           if(size%2!=0) // odd size inputList
Expected Return Value:
                                                                                                   8 +
                                                                                                   9
                                                                                                              int median_order = ((size+1)/2);
37.00
                                                                                                              res = (float)quick_select(inputList, start_index, end
                                                                                                  10
                                                                                                  11
                                                                                                  12
                                                                                                           else // even size inputList
Testcase 2:
                                                                                                  13 -
Input:
                                                                                                  14
                                                                                                              // Write code here
6,
                                                                                                  15
[-24, -16, -8, -4, -54, -1]
                                                                                                  16
                                                                                                           return res;
                                                                                                  17
                                                                                                  18
Expected Return Value:
                                                                                                  19
-12.00
```

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *printf()* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases.

Code Approach: For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method **manchester** print space-separated integers with the following property: for each element in the input list arr, if the bit arr[i] is the same as arr[i-1], then the element of the output list is 0. If they are different, then its 1. For the first bit in the input list, assume its previous bit to be 0. This encoding is stored in a new list.

The function/method **manchester** accepts two arguments - *len*, an integer representing the length of the list and *arr* and *arr*, a list of integers, respectively. Each element of *arr* represents a bit - 0 or 1

For example - if arr is {0 1 0 0 1 1 1 0}, the function/method should print an list {0 1 1 0 1 0 0 1}.

The function/method compiles successfully but fails to print the desired result for some test cases due to logical errors. Your task is to fix the code so that it passes all the test cases.

```
1  // You can print the values to stdout for debugging
2  void manchester(int len, int* arr)
3 * {
4    int* res = (int*)malloc(sizeof(int)*len);
5    res[0] = arr[0];
6 * for(int i = 1; i < len; i++){
7    res[i] = (arr[i]==arr[i-1]);
8    }
9    for(int i = 0; i < len; i++)
10         printf("%d ",res[i]);
11 }</pre>
```

```
// You can print the values to stdout for debugging
Testcase 1:
                                                                                                 void manchester(int len, int* arr)
                                                                                               3 + {
Input:
                                                                                                      int* res = (int*)malloc(sizeof(int)*len);
6, [1, 1, 0, 0, 1, 0]
                                                                                                      res[0] = arr[0];
                                                                                               6 *
                                                                                                      for(int i = 1; i < len; i++){
Expected Return Value:
                                                                                                        res[i] = (arr[i] == arr[i-1]);
101011
                                                                                               8
                                                                                               9
                                                                                                      for(int i =0; i<len; i++)
                                                                                              10
                                                                                                         printf("%d ",res[i]);
                                                                                              11 }
Testcase 2:
Input:
8, [0, 0, 0, 1, 0, 1, 1, 1]
Expected Return Value:
00011100
```

```
You are required to complete the given code. You can click on Compile &
Run anytime to check the compilation/execution status of the program. You can
                                                                                                  3 + {
use printf() to debug your code. The submitted code should be logically/syntactically
correct and pass all testcases. Do not write the main() function as it is not required.
Code Approach: For this question, you will need to complete the code as in given
implementation. We do not expect you to modify the approach.
You are given a predefined structure/class Point and also a collection of related
functions/methods that can be used to perform some basic operations on the structure.
The function/method isRightTriangle returns an integer '1', if the points make a right-
angled triangle otherwise return '0'.
The function/method isRightTriangle accepts three points - P1, P2, P3 representing the
input points.
You are supposed to use the given function to complete the code of the function/method
isRightTriangle so that it passes all test cases.
Helper Description
The following structure is used to represent point and is already implemented in the default
code (Do not write these definitions again in your code):
struct point;
typedef struct point
  int X;
 int Y;
}Point;
double Point calculateDistance(Point *point1, Point *point2)
```

```
int isRightTriangle(Point *P1, Point *P2, Point *P3)
   // write your code here
```

```
1 // You can print the values to stdout for debugging
Testcase 1:
                                                                                                   int isRightTriangle(Point *P1, Point *P2, Point *P3)
                                                                                                3 - {
Input:
                                                                                                       // write your code here
(2, -2),
(-2, -2),
(1, -2)
Expected Return Value:
0
Testcase 2:
Input:
(-1, 0),
(2, 0),
(-1, -4)
Expected Return Value:
```

You are required to fix all syntactical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *printf()* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the *main()* function as it is not required. **Code Approach**: For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method *multiplyNumber* returns an integer representing the multiplicative

The function/method *multiplyNumber* compiles unsuccessfully due to syntactical error. Your task is to debug the code so that it passes all the test cases.

product of the maximum two of three input numbers. The function/method multiplyNumber

accepts three integers- numA, numB and numC, representing the input numbers.

// You can print the values to stdout for debugging
int multiplyNumber(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;
}</pre>

10



```
You are required to fix all logical errors in the given code. You can click on Compile & Run anytime to check the compilation/execution status of the program. You can use printf() to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the main() function as it is not required.

Code Approach: For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

The function/method drawPrintPattern accepts num, an integer.

The function/method drawPrintPattern prints the first num lines of the pattern shown below.
```

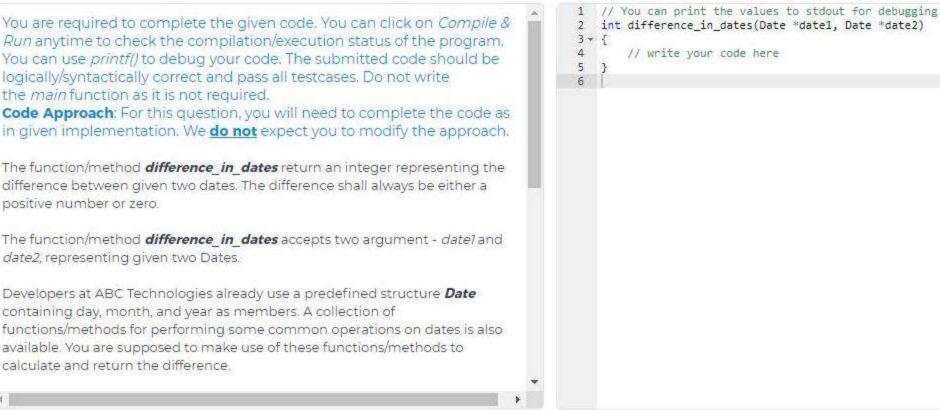
For example, if *num* = 3, the pattern should be:

```
1111
```

result for some test cases due to incorrect implementation of the function/method. Your task is to fix the code so that it passes all the test cases.

The function/method drawPrintPattern compiles successfully but fails to get the desired

```
// You can print the values to stdout for debugging
Testcase 1:
                                                                                                void drawPrintPattern(int num)
Input:
                                                                                             4
                                                                                                    int i,j,print = 1;
                                                                                                    for(i=1;i<=num;i++)
                                                                                             6 +
Expected Return Value:
                                                                                                        for(j=1;j<=2*i;j++)
1 1
                                                                                             8 +
1111
                                                                                                           printf("%d ",print);
                                                                                            10
111111
                                                                                            11
                                                                                                        printf("\n");
11111111
                                                                                            12
                                                                                            13
                                                                                            14
Testcase 2:
Input:
Expected Return Value:
1 1
```

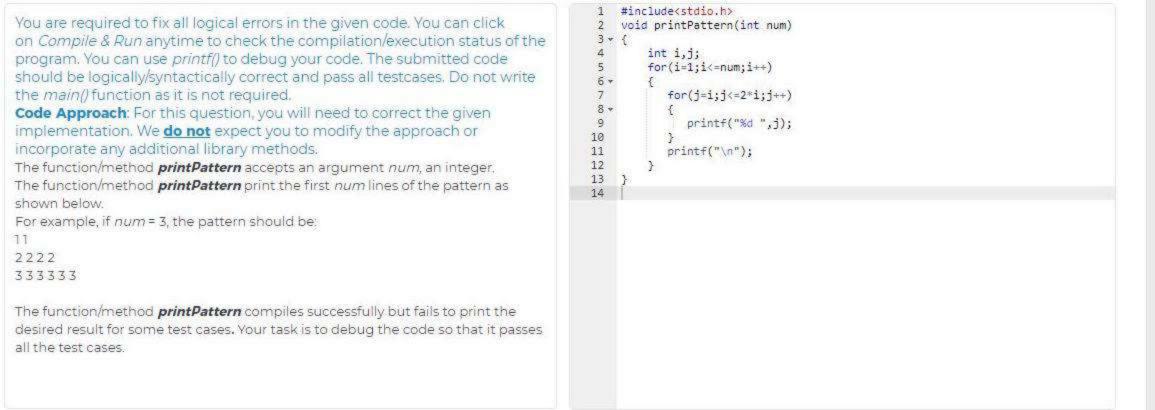


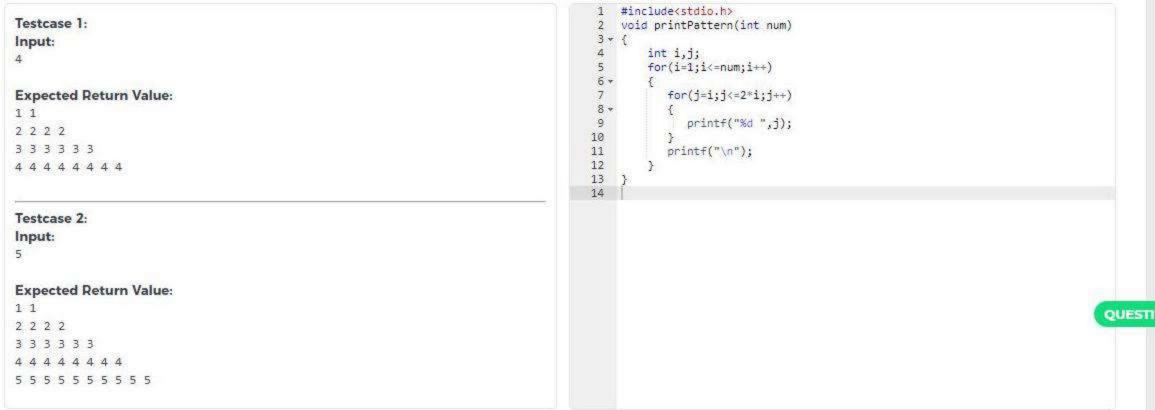
int difference in dates(Date *date1, Date *date2) // write your code here

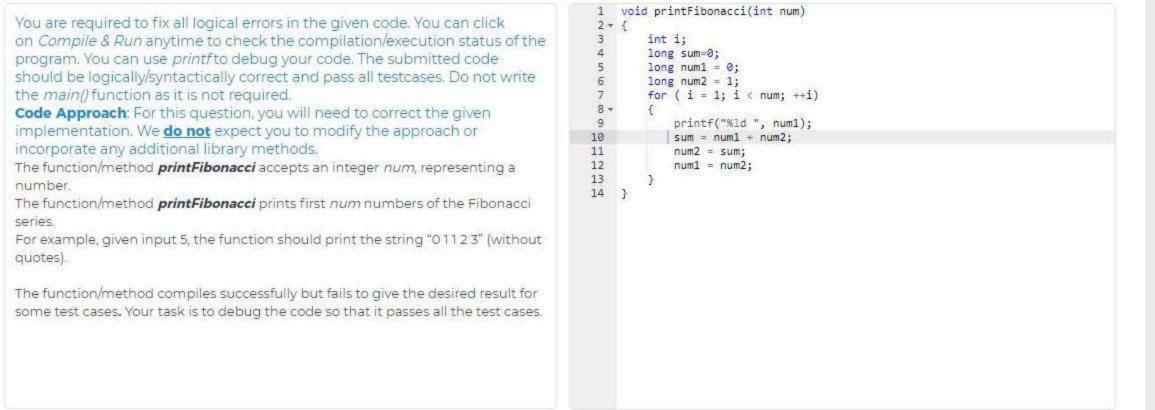


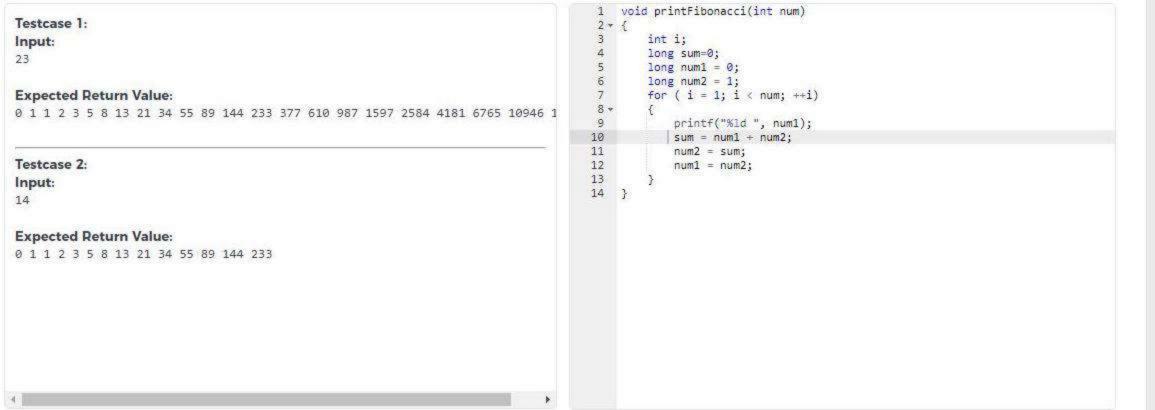
```
You are required to fix all syntactical errors in the given code. You can click
                                                                                           void replaceMinMax(int size, int* arr)
on Compile & Run anytime to check the compilation/execution status of the
                                                                                               int i:
program. You can use printf() to debug your code. The submitted code
                                                                                               if(size>0)
should be logically/syntactically correct and pass all testcases. Do not write
the main() function as it is not required.
                                                                                                   int max = arr.0:
                                                                                                   int min = arr.0;
Code Approach: For this question, you will need to correct the given
                                                                                                   for(i=0;i<size;++i)
implementation. We do not expect you to modify the approach or
incorporate any additional library methods.
                                                                                                       if(max<arr[i])
                                                                                      12 -
                                                                                                          max = arr[i];
The function/method replaceMinMax is supposed to replace all the even
                                                                                      14
elements of the input list with the maximum element of the list, also replace all
                                                                                      15
                                                                                                       else if(min > arr[i])
the odd elements of arr with the minimum element of the list.
                                                                                      16 -
                                                                                      17
                                                                                                           min = arr[i];
The function/method replaceMinMax accepts two arguments - size, an integer
                                                                                      19
representing the size of the input list and arr, a list of integers representing the
                                                                                                   for(i=0;i<size;++i)
                                                                                      21 -
input list.
                                                                                                       if(arr[i] % 2 == 0)
                                                                                      22
                                                                                                           arr[i]=max;
The function/method replaceMinMax compiles unsuccessfully due to syntactical
                                                                                      24
                                                                                                       else
error. Your task is to debug the code so that it passes all the test cases.
                                                                                                          arr[i]=min;
                                                                                      26
                                                                                      27
```

```
Testcase 1:
                                                                                             void replaceMinMax(int size, int* arr)
                                                                                          3 + 1
Input:
                                                                                                  int i:
                                                                                                  if(size>0)
[2, 5, 8, 11, 3]
                                                                                                      int max = arr.0;
                                                                                                      int min = arr.0;
Expected Return Value:
                                                                                                      for(i=0;i<size;++i)
11 2 11 2 2
                                                                                         10 -
                                                                                         11
                                                                                                          if(max<arr[i])
                                                                                         12 -
                                                                                         13
14
                                                                                                              max = arr[i];
Testcase 2:
Input:
                                                                                         15
16 +
                                                                                                          else if(min > arr[i])
9,
                                                                                         17
18
                                                                                                              min = arr[i];
[3, 2, 5, 8, 9, 11, 23, 45, 63]
                                                                                         19
Expected Return Value:
                                                                                                      for(i=0;i<size;++i)
2 63 2 63 2 2 2 2 2
                                                                                         21 -
                                                                                         22
                                                                                                          if(arr[i] % 2 == 0)
                                                                                         23
24
                                                                                                              arr[i]=max;
                                                                                                          else
                                                                                         25
26
27
                                                                                                              arr[i]=min;
                                                                                         28
```



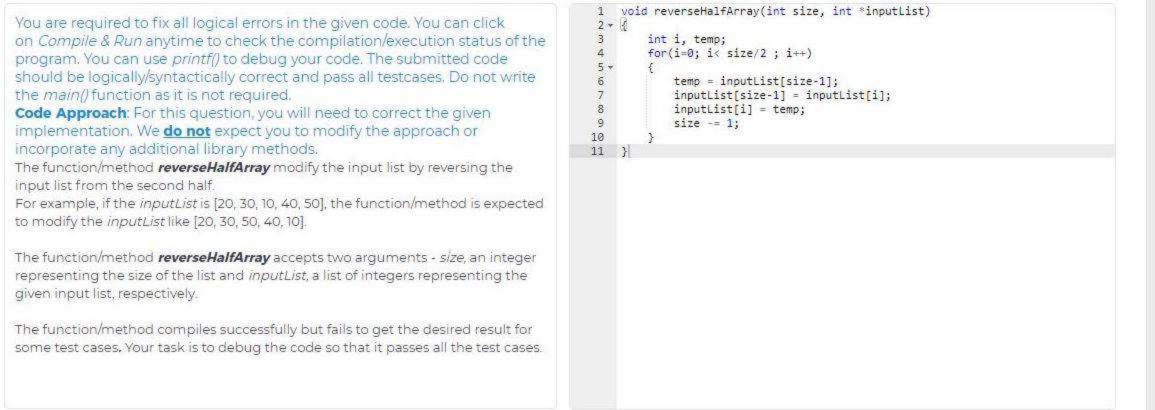


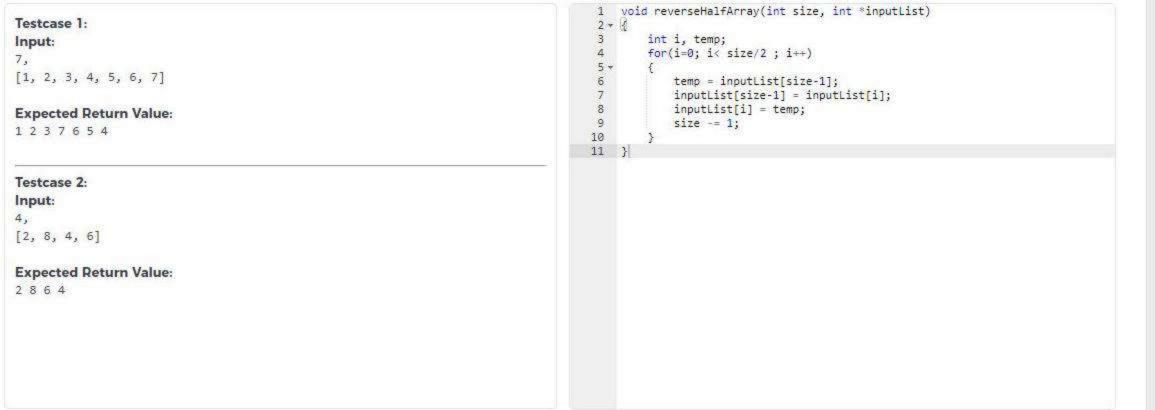




```
#include(stdbool.h>
You are required to fix all logical errors in the given code. You can click
                                                                                             void manchester(int size, int* arr)
on Compile & Run anytime to check the compilation/execution status of
                                                                                                 bool result:
the program. You can use printf to debug your code. The submitted code
                                                                                                 int* res = (int*)malloc(sizeof(int)*size);
should be logically/syntactically correct and pass all testcases. Do not
                                                                                                 int count =0;
write the main() function as it is not required.
                                                                                                 for(int i = 0; i < size; i++)
Code Approach: For this question, you will need to correct the given
                                                                                                     if(i==0)
implementation. We do not expect you to modify the approach or
                                                                                                         result= (arr[i]==0);
incorporate any additional library methods.
                                                                                                     else
                                                                                                         result = (arr[i]==arr[i-1]);
The function/method manchester print space-separated integers with the
                                                                                                     res[i] = (result)?(0):(++count);
following property: for each element in the input array arr, a counter is
                                                                                        14
incremented if the bit arrfil is the same as arrfi-1]. Then the increment counter
                                                                                        15
                                                                                                 for(int i=0; i<size; i++)
                                                                                        16 -
value is added to the output array to store the result.
                                                                                        17
                                                                                                     printf("%d ",res[i]);
If the bit arr[i] and arr[i-1] are different, then 0 is added to the output array. For
the first bit in the input array, assume its previous bit to be 0. For example, if
                                                                                        19
arr is [0,1,0,0,1,1,1,0], the function/method should print 1 0 0 2 0 3 4 0.
The function/method manchester accepts two arguments- size, an integer
representing the length of the input array, arr, a list of integers representing an
input array. Each element of arr represents a bit, 0 or 1.
The function/method manchester compiles successfully but fails to print the
desired result for some test cases due to logical errors. Your task is to fix the
code so that it passes all the test cases.
```

```
#include<stdbool.h>
Testcase 1:
                                                                                         void manchester(int size, int* arr)
                                                                                     3 +
Input:
                                                                                             bool result;
6,
                                                                                             int* res = (int*)malloc(sizeof(int)*size);
[1, 1, 0, 0, 1, 0]
                                                                                             int count =0;
                                                                                             for(int i = 0; i < size; i++)
Expected Return Value:
                                                                                                 if(i==0)
010200
                                                                                     10
                                                                                                    result= (arr[i]==0);
                                                                                                 else
                                                                                    12
                                                                                                    result = (arr[i]==arr[i-1]);
                                                                                    13
14
                                                                                                 res[i] = (result)?(0):(++count);
Testcase 2:
Input:
                                                                                    15
16 +
                                                                                             for(int i=0; i<size; i++)
8,
                                                                                    17
18
                                                                                                 printf("%d ",res[i]);
[0, 0, 0, 1, 0, 1, 1, 1]
                                                                                    19
Expected Return Value:
12300045
```





// You can print the values to stdout for debugging You are required to complete the given code. You can click on Compile & int isRightTriangle(Point *P1, Point *P2, Point *P3) Run anytime to check the compilation/execution status of the program. // write your code here You can use printf/) to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the main() function as it is not required. Code Approach: For this question, you will need to complete the code as in given implementation. We do not expect you to modify the approach. You are given a predefined structure/class Point and also a collection of related functions/methods that can be used to perform some basic operations on the structure. The function/method isRightTriangle returns an integer '1', if the points make a right-angled triangle otherwise return '0'. The function/method isRightTriangle accepts three points - P1, P2, P3 representing the input points. You are supposed to use the given function to complete the code of the function/method isRightTriangle so that it passes all test cases. Helper Description The following structure is used to represent point and is already implemented in the default code (Do not write these definitions again in your code): struct point;











You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *printf()* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases.

Code Approach: For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

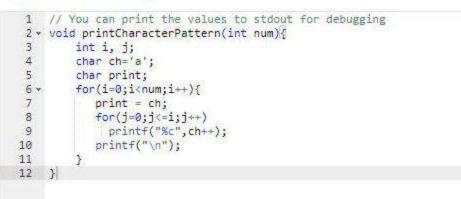
The function/method **printCharacterPattern** accepts an integer *num*. It is supposed to print the first num ($0 \le num \le 26$) lines of the pattern as shown below.

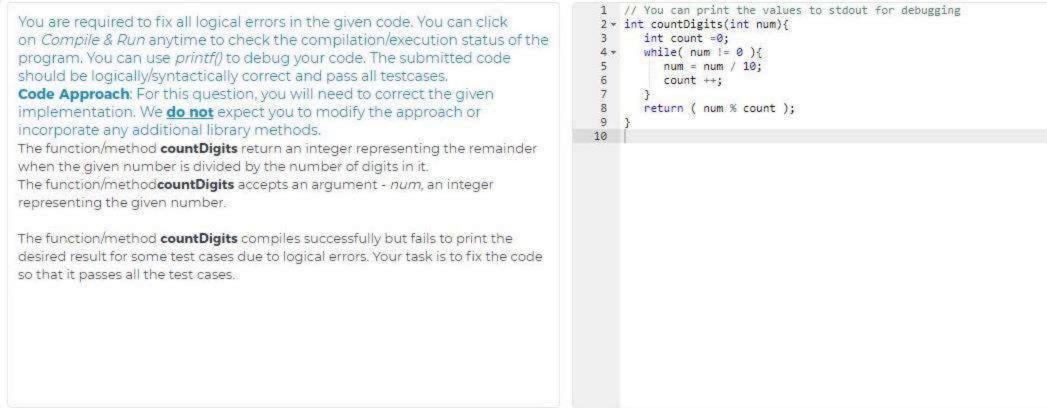
For example, if *num* = 4, the pattern is: a ab

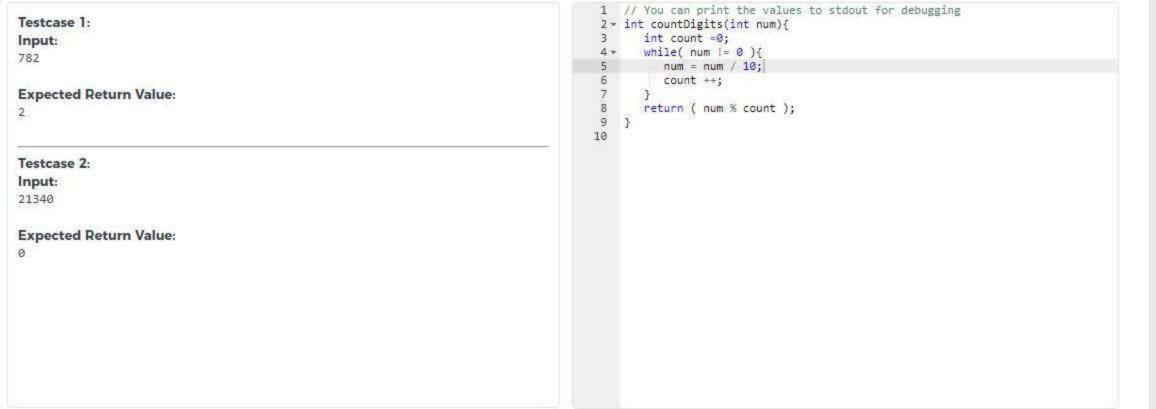
abc abcd

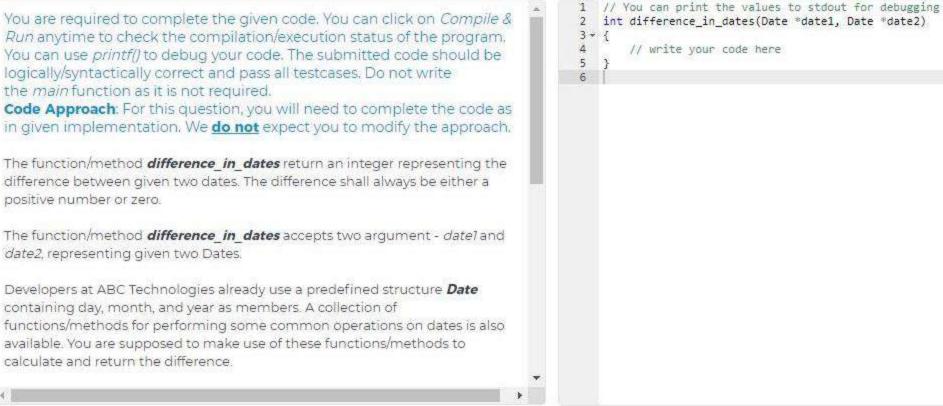
Problem

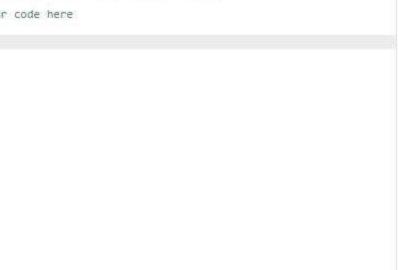
The function/method compiles successfully but fails to print the desired result for some test cases due to logical errors. Your task is to fix the code so that it passes all the test cases.





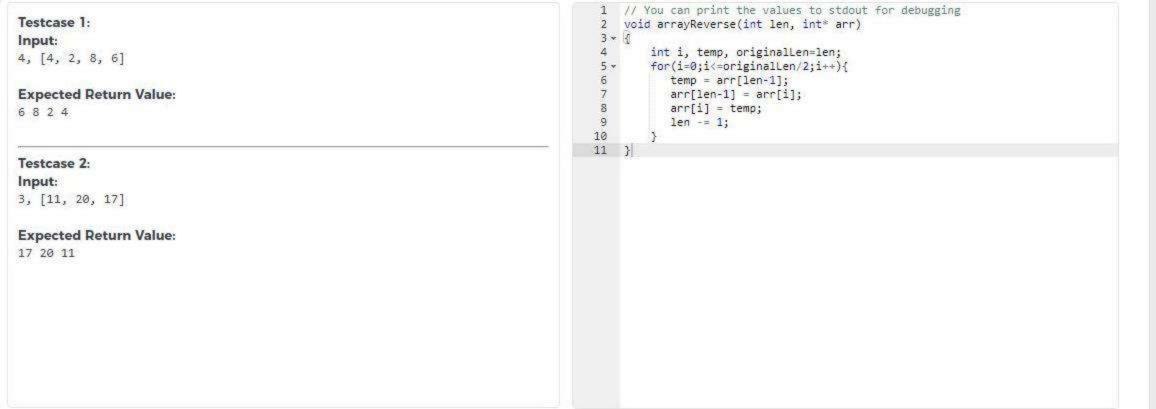






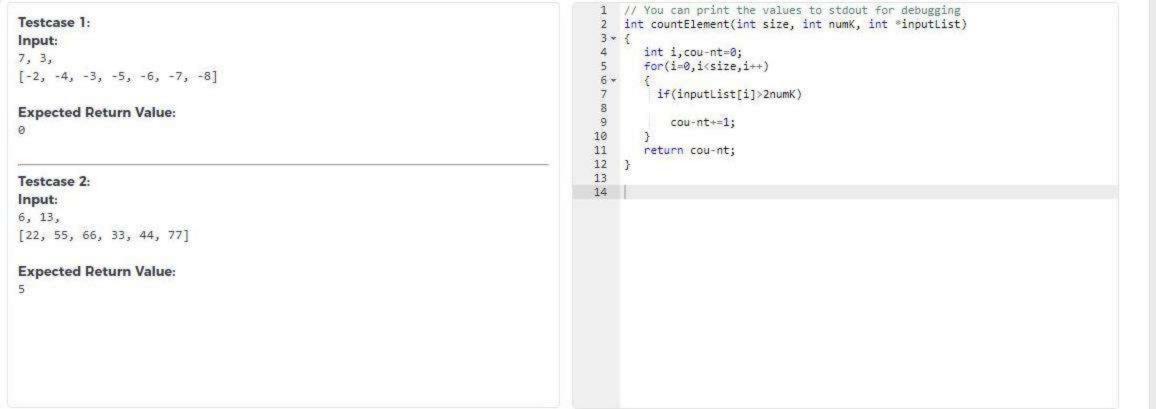
Testcase 1: Input:	<pre>1 // You can print the values to stdout for debugging 2 int difference_in_dates(Date *date1, Date *date2) 3 * {</pre>
{day:02, month:05, year:2013},	3 + { 4 // write your code here
{day:02, month:06, year:2013}	5 }
Expected Return Value:	
31	
Testcase 2:	
Input:	
{day:01, month:06, year:2011},	
{day:01, month:06, year:2012}	
Expected Return Value:	
366	

```
You are required to fix all logical errors in the given code. You can click
                                                                                           void arrayReverse(int len, int* arr)
on Compile & Run anytime to check the compilation/execution status of the
                                                                                        3 + {
                                                                                               int i, temp, originalLen=len;
program. You can use printf() to debug your code. The submitted code
                                                                                               for(i=0;i<=originalLen/2;i++){
should be logically/syntactically correct and pass all testcases,
                                                                                                  temp = arr[len-1];
Code Approach: For this question, you will need to correct the given
                                                                                                  arr[len-1] = arr[i];
                                                                                                  arr[i] = temp;
implementation. We do not expect you to modify the approach or
                                                                                                  len -= 1:
incorporate any additional library methods.
The function/method arrayReverse modify the input list by reversing its element
The function/method arrayReverse accepts two arguments - len, an integer
representing the length of the list and arr, list of integers representing the input
list, respectively.
For example, if the input list arr is [20 30 10 40 50], the function/method is
supposed to print [50 40 10 30 20].
The function/method arrayReverse compiles successfully but fails to get the
desired result for some test cases due to logical errors. Your task is to fix the code
so that it passes all the test cases.
```

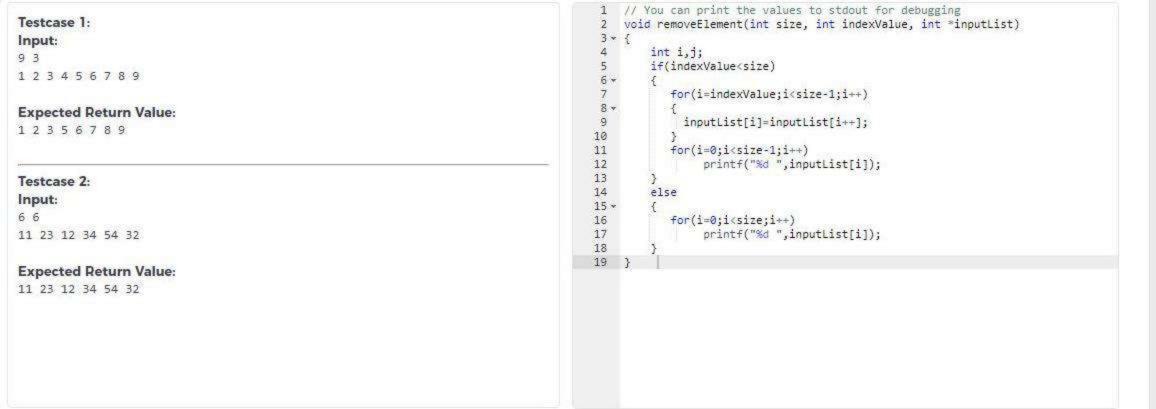


```
// You can print the values to stdout for debugging
You are required to fix all syntactical errors in the given code. You can click
                                                                                          int countElement(int size, int numK, int *inputList)
on Compile & Run anytime to check the compilation/execution status of the
                                                                                             int i,cou-nt=0;
program. You can use print to debug your code. The submitted code should
                                                                                             for(i=0,i<size,i++)
be logically/syntactically correct and pass all testcases. Do not write
the main() function as it is not required.
                                                                                               if(inputList[i]>2numK)
Code Approach: For this question, you will need to correct the given
                                                                                                 cou-nt+=1;
implementation. We do not expect you to modify the approach or
incorporate any additional library methods.
                                                                                             return cou-nt;
                                                                                      12
The function/method countElement return an integer representing the number
                                                                                      14
of elements in the input list which are greater than twice the input number K.
The function/method countElement accepts three arguments - size, an integer
representing the size of the input list, numK, an integer representing the input
number K and inputList, a list of integers representing the input list, respectively.
The function/method countElement compiles unsuccessfully due to syntactical
error. Your task is to fix the code so that it passes all the test cases.
```

inputList)



```
You can print the values to stdout for debugging
You are required to fix all logical errors in the given code. You can click
                                                                                          void removeElement(int size, int indexValue, int *inputList)
on Compile & Run anytime to check the compilation/execution status of
                                                                                              int i,j;
the program. You can use printf() to debug your code. The submitted
                                                                                              if(indexValue<size)
code should be logically/syntactically correct and pass all testcases. Do
not write the main() function as it is not required.
                                                                                                 for(i=indexValue;i<size-1;i++)
Code Approach: For this question, you will need to correct the given
                                                                                                   inputList[i]=inputList[i++];
implementation. We do not expect you to modify the approach or
incorporate any additional library methods.
                                                                                                 for(i=0;i<size-1;i++)
The function/method removeElement prints space separated integers that
                                                                                                      printf("%d ",inputList[i]);
remains after removing the integer at the given index from the input list.
                                                                                              else
                                                                                      15 -
The function/method removeElement accepts three arguments - size, an
                                                                                      16
                                                                                                 for(i=0:i<size:i++)
                                                                                                      printf("%d ",inputList[i]);
integer representing the size of the input list, indexValue, an integer
representing given index and inputList, a list of integers representing the input
                                                                                      19
list
The function/method removeElement compiles successfully but fails to print
the desired result for some test cases due to incorrect implementation of the
function/method removeElement Your task is to fix the code so that it passes
all the test cases.
Note:
```



```
You can print the values to stdout for debugging
You are required to complete the given code, You can click on Compile &
                                                                                           int* sortArray(int len, int* arr)
Run anytime to check the compilation/execution status of the program.
                                                                                               int i=0, j=0, temp=0;
You can use printf() to debug your code. The submitted code should be
                                                                                               for(i=0;i<len;i++)
logically/syntactically correct and pass all testcases. Do not write
the main function as it is not required.
Code Approach: For this question, you will need to complete the code as
in given implementation. We do not expect you to modify the approach.
                                                                                       10 -
The function/method findMaxElement return an integer representing the
largest element in the given two input lists.
The function/method findMaxElement accepts four arguments - len1, an
integer representing the length of the first list, arr1, a list of integers
                                                                                               return arr:
representing the first input list, len2, an integer representing the length of the
second input list and arr2, a list of integers representing the second input list,
respectively.
                                                                                       21 - {
                                                                                               // write your code here
Another function/method sortArray accepts two arguments - len, an integer
representing the length of the list and arr, a list of integers, respectively and
                                                                                       24
return a list sorted ascending order.
Your task is to use the function/method sortArray to complete the code in
findMaxElement so that it passes all the test cases.
```

```
int findMaxElement(int len1, int* arr1, int len2, int* arr2)
```

for(j=i+1;j<len;j++)

if(arr[i]>arr[j])

temp = arr[i]; arr[i] = arr[j];

arr[j] = temp;

```
Testcase 1:
                                                                                           int* sortArray(int len, int* arr)
                                                                                        3 + {
Input:
                                                                                               int i=0, j=0, temp=0;
12, [2, 5, 1, 3, 9, 8, 4, 6, 5, 2, 3, 11],
                                                                                               for(i=0;i<len;i++)
11, [11, 13, 2, 4, 15, 17, 67, 44, 2, 100, 23]
                                                                                                   for(j=i+1;j<len;j++)
Expected Return Value:
                                                                                                       if(arr[i]>arr[j])
100
                                                                                      10 -
11
                                                                                                           temp = arr[i];
                                                                                                           arr[i] = arr[j];
                                                                                                          arr[j] = temp;
Testcase 2:
                                                                                      14
15
16
Input:
7, [100, 22, 43, 912, 56, 89, 85]
                                                                                               return arr;
6, [234, 123, 456, 234, 890, 101]
Expected Return Value:
                                                                                           int findMaxElement(int len1, int* arr1, int len2, int* arr2)
912
                                                                                      21 - {
                                                                                      22
                                                                                               // write your code here
                                                                                      23
                                                                                      24
```

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *printf()* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases.

Code Approach: For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library

methods.

The function/method *manchester* print space-separated integers with the following property:

for each element in the input list *arr*, if the bit arr[i] is the same as *arr*[i-1], then the element of the output list is 0. If they are different, then its 1. For the first bit in the input list, assume its previous bit to be 0. This encoding is stored in a new list.

The function/method **manchester** accepts two arguments - *len*, an integer representing the length of the list and *arr* and *arr*, a list of integers, respectively. Each element of *arr* represents a bit - 0 or 1

For example - if arr is {01001110}, the function/method should print an list {01101001}.

cases due to logical errors. Your task is to fix the code so that it passes all the test cases.

The function/method compiles successfully but fails to print the desired result for some test

```
1  // You can print the values to stdout for debugging
2  void manchester(int len, int* arr)
3 * {
4    int* res = (int*)malloc(sizeof(int)*len);
5    res[0] = arr[0];
6 * for(int i = 1; i < len; i++){
7    res[i] = (arr[i]==arr[i-1]);
8    }
9    for(int i = 0; i < len; i++)
10         printf("%d ",res[i]);
11 }</pre>
```

You are required to fix all syntactical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *printf()* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the *main()* function as it is not required.

Code Approach: For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

The function/method *multiplyNumber* returns an integer representing the multiplicative

accepts three integers- numA, numB and numC, representing the input numbers.

The function/method *multiplyNumber* compiles unsuccessfully due to syntactical error. Your task is to debug the code so that it passes all the test cases.

product of the maximum two of three input numbers. The function/method multiplyNumber

```
// You can print the values to stdout for debugging
int multiplyNumber(int numA, int numB, int numC)

int result,min,max,mid;
  max=(numA>numB)?numA>numC)?numA:numC):(numB>numC)?numB:num
  min=(numA<numB)?((numA<numC)?numA:numC):((numB<numC)?numB
  mid=(numA+numB+numC)-(min+max);
  result=(max*int mid);
  return result;
}</pre>
```

10

```
You are required to complete the given code. You can click on Compile & Run anytime
to check the compilation/execution status of the program. You can use printf() to debug
your code. The submitted code should be logically/syntactically correct and pass
all testcases. Do not write the main() function as it is not required.
Code Approach: For this question, you will need to complete the code as in given
implementation. We do not expect you to modify the approach.
The function/method median accepts two arguments - size and inputList, an integer
representing the length of a list and a list of integers, respectively.
The function/method median is supposed to calculate and return an integer representing the
median of elements in the input list. However, the function/method median works only for
odd-length lists because of incomplete code.
You must complete the code to make it work for even-length lists as well. A couple of other
functions/methods are available, which you are supposed to use inside the function/method
median to complete the code.
Helper Description
The following function is used to represent a quick_select and is already implemented in the
default code (Do not write this definition again in your code):
int quick select(int* inputList, int start index, int end index, int median order)
   /*It calculate the median value
   This can be called as -
   quick select(inputList, start index, end index, median order)
   where median order is the half length of the inputList
```

```
// You can print the values to stdout for debugging
    float median(int size, int * inputList)
3 +
        int start index = 0;
        int end index = size-1;
        float res = -1:
        if(size%2!=0) // odd size inputList
           int median order = ((size+1)/2);
           res = (float)quick select(inputList, start index, end
        else // even size inputList
13 -
            // Write code here
        return res;
```

10

11 12

14

15

16

17

18

19

```
Run anytime to check the compilation/execution status of the program. You can
use printf() to debug your code. The submitted code should be logically/syntactically
correct and pass all testcases. Do not write the main() function as it is not required.
                                                                                                  6
Code Approach: For this question, you will need to complete the code as in given
implementation. We do not expect you to modify the approach.
You are given a predefined structure/class Point and also a collection of related
functions/methods that can be used to perform some basic operations on the structure.
The function/method isRightTriangle returns an integer '1', if the points make a right-
angled triangle otherwise return '0'.
The function/method isRightTriangle accepts three points - P1, P2, P3 representing the
input points.
You are supposed to use the given function to complete the code of the function/method
isRightTriangle so that it passes all test cases.
Helper Description
The following structure is used to represent point and is already implemented in the default
code (Do not write these definitions again in your code):
struct point;
typedef struct point
 int X;
 int Y;
}Point;
double Point calculateDistance(Point *point1, Point *point2)
```

You are required to complete the given code. You can click on Compile &

```
// You can print the values to stdout for debugging
int isRightTriangle(Point *P1, Point *P2, Point *P3)
{
    // Write your code here
}
```

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *printf()* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases.

Code Approach: For this question, you will need to correct the given implementation.

We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method **countOccurrence** return an integer representing the count of occurrences of given value in the input list.

The function/method *countOccurrence* accepts three arguments - *len*, an integer representing the size of the input list, *value*, an integer representing the given value and *arr*, a list of integers, representing the input list.

The function/method *countOccurrence* compiles successfully but fails to return the desired

result for some test cases due to logical errors. Your task is to fix the code so that it passes all the test cases.

```
1  // You can print the values to stdout for debugging
2  int countOccurrence( int len, int value, int *arr)
3 * {
4    int i=0, count = 0;
    while(i<len){
        if(arr[i]==value)
            count += 1;
        }
        return count;
10  }
11</pre>
```

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *printf()* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the *main()* function as it is not required. **Code Approach**: For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method *drawPrintPattern* accepts *num*, an integer.

The function/method *drawPrintPattern* prints the first *num* lines of the pattern shown below.

For example, if *num* = 3, the pattern should be:

1111

result for some test cases due to incorrect implementation of the function/method. Your task is to fix the code so that it passes all the test cases.

The function/method drawPrintPattern compiles successfully but fails to get the desired

You are required to fix all the logical error in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *printf()* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the *main()* function as it is not required.

Code Approach: For this question, you will need to complete the code as in given implementation. We **do not** expect you to modify the approach.

The function/method *sameElementCount* returns an integer representing the number of elements of the input list which are even numbers and equal to the element to its right. For example, if the input list is [4 4 4 1 8 4 1 1 2 2] then the function/method should return the

The function/method **sameElementCount** accepts two arguments - *size*, an 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

output '3' as it has three similar groups i.e, (4, 4), (4, 4), (2, 2).

cases due to incorrect implementation of the function/method **sameElementCount**. Your task is to fix the code so that it passes all the test cases.

Note: In a list, an element at index i is considered to be on the left of index i+1 and to the right of

index i-1. The last element of the input list does not have any element next to it which makes it incapable to satisfy the second condition and hence should not be counted.

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

10

11

12

13

```
You are required to complete the given code. You can click on Compile & Run anytime
to check the compilation/execution status of the program. You can use printf() to debug
your code. The submitted code should be logically/syntactically correct and pass
all testcases. Do not write the main() function as it is not required.
Code Approach: For this question, you will need to complete the code as in given
implementation. We do not expect you to modify the approach.
The function/method median accepts two arguments - size and inputList, an integer
                                                                                                 10
representing the length of a list and a list of integers, respectively.
                                                                                                 11
                                                                                                 12
The function/method median is supposed to calculate and return an integer representing the
                                                                                                 13 -
median of elements in the input list. However, the function/method median works only for
                                                                                                 14
                                                                                                 15
odd-length lists because of incomplete code.
                                                                                                 16
                                                                                                 17
You must complete the code to make it work for even-length lists as well. A couple of other
                                                                                                 18
functions/methods are available, which you are supposed to use inside the function/method
                                                                                                 19
median to complete the code.
Helper Description
The following function is used to represent a quick_select and is already implemented in the
default code (Do not write this definition again in your code):
int quick select(int* inputList, int start index, int end index, int median order)
   /*It calculate the median value
   This can be called as -
   quick select(inputList, start index, end index, median order)
   where median order is the half length of the inputList
```

```
float median(int size, int * inputList)
    int start index = 0;
    int end index = size-1;
    float res = -1;
    if(size%2!=0) // odd size inputList
       int median order = ((size+1)/2);
       res = (float)quick select(inputList, start index, end
    else // even size inputList
        // Write code here
    return res;
```

```
Testcase 1:
                                                                                                       float median(int size, int * inputList)
Input:
                                                                                                           int start index = 0;
5,
                                                                                                           int end index = size-1;
[2, 40, 23, 52, 37]
                                                                                                           float res = -1;
                                                                                                           if(size%2!=0) // odd size inputList
Expected Return Value:
                                                                                                    8 -
                                                                                                              int median order = ((size+1)/2);
37.00
                                                                                                   10
                                                                                                              res = (float)quick_select(inputList, start_index, end
                                                                                                   11
                                                                                                   12
                                                                                                           else // even size inputList
Testcase 2:
                                                                                                   13 -
14
Input:
                                                                                                               // Write code here
                                                                                                   15
6,
[-24, -16, -8, -4, -54, -1]
                                                                                                   16
                                                                                                           return res;
                                                                                                   17
                                                                                                   18
Expected Return Value:
                                                                                                   19
-12.00
```

```
correct and pass all testcases. Do not write the main() function as it is not required.
                                                                                                  3 + {
Code Approach: For this question, you will need to complete the code as in given
implementation. We do not expect you to modify the approach.
You are given a predefined structure/class Point and also a collection of related
functions/methods that can be used to perform some basic operations on the structure.
The function/method isRightTriangle returns an integer '1', if the points make a right-
angled triangle otherwise return '0'.
The function/method isRightTriangle accepts three points - P1, P2, P3 representing the
input points.
You are supposed to use the given function to complete the code of the function/method
isRightTriangle so that it passes all test cases.
Helper Description
The following structure is used to represent point and is already implemented in the default
code (Do not write these definitions again in your code):
struct point;
typedef struct point
  int X;
  int Y;
}Point;
double Point calculateDistance(Point *point1, Point *point2)
    /* Return the distance between point1 and point2;
      This can be called as -
```

use printf() to debug your code. The submitted code should be logically/syntactically

```
int isRightTriangle(Point *P1, Point *P2, Point *P3)
   // write your code here
```

```
You are required to complete the given code. You can click on Compile &
Run anytime to check the compilation/execution status of the program. You can
                                                                                                  3 - {
use printf() to debug your code. The submitted code should be logically/syntactically
correct and pass all testcases. Do not write the main() function as it is not required.
Code Approach: For this question, you will need to complete the code as in given
implementation. We do not expect you to modify the approach.
You are given a predefined structure/class Point and also a collection of related
functions/methods that can be used to perform some basic operations on the structure.
The function/method isRightTriangle returns an integer '1', if the points make a right-
angled triangle otherwise return '0'.
The function/method isRightTriangle accepts three points - P1, P2, P3 representing the
input points.
You are supposed to use the given function to complete the code of the function/method
isRightTriangle so that it passes all test cases.
Helper Description
The following structure is used to represent point and is already implemented in the default
code (Do not write these definitions again in your code):
struct point;
typedef struct point
  int X;
 int Y;
}Point;
double Point calculateDistance(Point *point1, Point *point2)
```

```
int isRightTriangle(Point *P1, Point *P2, Point *P3)
   // write your code here
```

You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *printf()* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases.

Code Approach: For this question, you will need to correct the given implementation.

We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method *countOccurrence* return an integer representing the count of occurrences of given value in the input list.

The function/method **countOccurrence** accepts three arguments - *len*, an integer representing the size of the input list, *value*, an integer representing the given value and *arr*, a list of integers, representing the input list.

The function/method *countOccurrence* compiles successfully but fails to return the desired

result for some test cases due to logical errors. Your task is to fix the code so that it passes all the test cases.

```
1  // You can print the values to stdout for debugging
2  int countOccurrence( int len, int value, int *arr)
3 * {
4    int i=0, count = 0;
5 * while(i<len){
6    if(arr[i]==value)
7    count += 1;
8    }
9    return count;
10 }</pre>
```

```
1 // You can print the values to stdout for debugging
Testcase 1:
                                                                                                    int countOccurrence( int len, int value, int *arr)
Input:
7, 3, [2, 3, 4, 3, 5, 6, 7]
                                                                                                       int i=0, count = 0;
                                                                                                 5 -
                                                                                                       while(i<len){
                                                                                                          if(arr[i]==value)
Expected Return Value:
                                                                                                             count += 1;
                                                                                                 8
                                                                                                9
                                                                                                       return count;
                                                                                                11
Testcase 2:
Input:
1, 2, [9]
Expected Return Value:
```

```
You are required to fix all logical errors in the given code. You can click on Compile & Run anytime to check the compilation/execution status of the program. You can use printf() to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the main() function as it is not required.

Code Approach: For this question, you will need to correct the given implementation. We do not expect you to modify the approach or incorporate any additional library methods.

The function/method drawPrintPattern accepts num, an integer.

The function/method drawPrintPattern prints the first num lines of the pattern shown below.

For example, if num = 3, the pattern should be:
```

11

```
111111
```

result for some test cases due to incorrect implementation of the function/method. Your task is to fix the code so that it passes all the test cases.

The function/method drawPrintPattern compiles successfully but fails to get the desired

```
// You can print the values to stdout for debugging
    void drawPrintPattern(int num)
        int i,j,print = 1;
        for(i=1;i<=num;i++)
            for(j=1;j<=2*i;j++);
                printf("%d ",print);
10
11
            printf("\n");
12
13
14
```

Run anytime to check the compilation/execution status of the program. You can use printf() to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the main() function as it is not required.

Code Approach: For this question, you will need to complete the code as in given implementation. We do not expect you to modify the approach.

The function/method sameElementCount returns an integer representing the number of elements of the input list which are even numbers and equal to the element to its right. For example, if the input list is [4 4 4 18 4 11 2 2] then the function/method should return the output '3' as it has three similar groups i.e, (4, 4), (4, 4), (2, 2).

The function/method sameElementCount accepts two arguments - size, an integer representing the size of the input list and inputList, a list of integers representing the input list.

You are required to fix all the logical error in the given code. You can click on Compile &

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

Note:In a list, an element at index i is considered to be on the left of index i+1 and to the right of

index i-1. The last element of the input list does not have any element next to it which makes it incapable to satisfy the second condition and hence should not be counted.

10

11

12

13



You are required to fix all logical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *printf()* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases.

Code Approach: For this question, you will need to correct the given implementation.

We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method **manchester** print space-separated integers with the following property: for each element in the input list arr, if the bit arr[i] is the same as arr[i-1], then the element of the output list is 0. If they are different, then its 1. For the first bit in the input list, assume its previous bit to be 0. This encoding is stored in a new list.

The function/method **manchester** accepts two arguments - *len*, an integer representing the length of the list and *arr* and *arr*, a list of integers, respectively. Each element of *arr* represents a bit - 0 or 1

For example - if arr is {01001110}, the function/method should print an list {01101001}.

The function/method compiles successfully but fails to print the desired result for some test

cases due to logical errors. Your task is to fix the code so that it passes all the test cases.

```
1  // You can print the values to stdout for debugging
2  void manchester(int len, int* arr)
3 * {
4    int* res = (int*)malloc(sizeof(int)*len);
5    res[0] = arr[0];
6 * for(int i = 1; i < len; i++){
7    res[i] = (arr[i]==arr[i-1]);
8    }
9    for(int i = 0; i < len; i++)
10         printf("%d ",res[i]);
11 }</pre>
```

```
1 // You can print the values to stdout for debugging
Testcase 1:
                                                                                                  void manchester(int len, int* arr)
                                                                                               3 - {
Input:
6, [1, 1, 0, 0, 1, 0]
                                                                                                      int* res = (int*)malloc(sizeof(int)*len);
                                                                                                      res[0] = arr[0];
                                                                                               6 =
                                                                                                      for(int i = 1; i < len; i++){
Expected Return Value:
                                                                                                        res[i] = (arr[i] == arr[i-1]);
101011
                                                                                               8
                                                                                              9
                                                                                                      for(int i =0; i<len; i++)
                                                                                                         printf("%d ",res[i]);
                                                                                              11 }
Testcase 2:
Input:
8, [0, 0, 0, 1, 0, 1, 1, 1]
Expected Return Value:
00011100
```

You are required to fix all syntactical errors in the given code. You can click on *Compile & Run* anytime to check the compilation/execution status of the program. You can use *printf()* to debug your code. The submitted code should be logically/syntactically correct and pass all testcases. Do not write the *main()* function as it is not required. **Code Approach**: For this question, you will need to correct the given implementation. We **do not** expect you to modify the approach or incorporate any additional library methods.

The function/method *multiplyNumber* 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.

The function/method *multiplyNumber* compiles unsuccessfully due to syntactical error. Your task is to debug the code so that it passes all the test cases.

```
// You can print the values to stdout for debugging
int multiplyNumber(int numA, int numB, int numC)

int result,min,max,mid;
  max=(numA>numB)?numA>numC)?numA:numC):(numB>numC)?numB:numin=(numA<numB)?((numA<numC)?numA:numC):((numB<numC)?numB
mid=(numA+numB+numC)-(min+max);
  result=(max*int mid);
  return result;</pre>
```

10

