7 (b) CTS Test 7

Test Summary

- No. of Sections: 1
- No. of Questions: 7
- Total Duration: 30 min

Section 1 - Automata Fix

Section Summary

- No. of Questions: 7
- Duration: 30 min

Additional Instructions:
None

Q1. 1. Lisa always forgets her birthday which is on the 5 th July

In order to help her, we have a class **BirthDay** having a method **checkBirthDay(String month,it day)** which takes day and month as inputs and return 1Yesif it is her birthday else return No

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

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

Test case 1:

Input:
July
13

Expected return value:
No

Test case 2:

Input:
April
3

Expcted return value:
No

PROGRAM

```
Public class BirthDay
{
Public static int checkBirthdayDay(String month, int day)
{
if(!(month==”July”)||(day!=5))
return 1;
else
return 0;
}
}
```

Sample Input

```
july
5
```

Sample Output

```
Yes
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q2. 1. The method **removeElement(int arr[],int element)** of class **ShortArray** takes an array arr as an input.It is supposed to return an array removing the integer if it is present in the input array arr. If the given integer is not in the array, then this function should r the input array arr.

The function compiles successfully but fails to return the desired result due to logical erros

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

Assumptions
The input index is always a non negative integer.
Zero based indexing is followed to access array elements.

Test case 1:

Input:
[1,2,3,4,5,6,7,8,9],3

Expected return value:
[1,2,3,5,6,7,8,9]

Test case 2:

Input:
[11,23,12,34,54,32],6

Expcted return value:
[11,23,12,34,54,32]

PROGRAM:

```
public class ShortArray{
public static int[] removeElement(int arr[],int n,int x){
if (arr[n-1] == x)
return (n-1);
int prev = arr[n-1], i;
for (i=n-2; i>=0 && arr[i]!=x; i--)
{
int curr = arr[i];
arr[i] = prev;
prev = curr;
}
if (i < 0)
return 0;
arr[i] = prev;

return (n-1); }
}
```

Sample Input

```
5
1 2 3 4 5
3
```

Sample Output

```
1 2 4 5
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q3. **QUESTION**

The function Manchester(int *arr,int len)accepts an array arr of legth len (len>0) as an input.Each element of an array represents a bit 0 or 1.The output is an array with the following property.

PROGRAM:

```
#include<stdio.h>
void Manchester(int arr, int len){
for(int i= 1; i< len-1; i++){
 if(arr[i]==arr[i-1])
   res[i]=1;
 else
   res[i]=0;
}
 for(int i=1;i<len;i++)
      printf("%d ",res[i]);
}
int main()
{
   int arr[100],i,len;
   scanf("%d",&len);
   for(i=0;i<len;i++)
      scanf("%d",&arr[i]);
   Manchester(arr,len);
   return 0;
}
```

Sample Input

```
4
1 1 2 2
```

Sample Output

```
1 0 1
```

Sample Input

```
5
1 1 1 1 2
```

Sample Output

```
1 1 1 0
```

Time Limit: 50 ms Memory Limit: 256 kb Code Size: 256 kb

Q4. **QUESTION:**

The method median(int arr[]) of class Median accepts an integer array arr. It is supposed to calculate and return the median of elements in the input array.

However, incomplete code in the method median (int arr[]) works only for odd length arrays.

```
#include <stdio.h>
void Array_sort(int *array , int n)
{
  int i=0 , j=0 , temp=0;
  for(i=0 ; i<n ; i++)
  {
     for(j=0 ; j<n-1 ; j++)
     {
        if(array[j]<array[j+1])
        {
           temp      = array[j];
           array[j]  = array[j+1];
           array[j+1] = temp;
        }
     }
  }
}
float Find_median(int array[] , int n)
{
  float median=0;
  if((n/2)!=1)
     median = n/2;
  return median;
}
int main()
{
  int array_1[30] = {0};
  int i=0 ,n=0;
  float median=0;
  scanf("%d",&n);
  for(i=0 ; i<n ; i++)
  {
     scanf("%d",&array_1[i]);
  }
  Array_sort(array_1 , n);
  median = Find_median(array_1 , n);
  printf("%f",median);
  return 0;
}
```

Sample Input

```
4
1 2 7 8
```

Sample Output

```
2.000000
```

Sample Input

```
6
10 20 100 45 201 459
```

Sample Output

```
45.000000
```

Time Limit: 50 ms Memory Limit: 256 kb Code Size: 256 kb

Q5. The function **maxReplace (int *arr,int len)** is supposed to replace every element of the input array arr of length len,with the maximum element of arr. The function looks fine but gives a compilation error.Your task is to fix the program so that it passes all test cases.

```
Int* maxReplace(int &amp;arr, int len)
{
Int i;
If(len&gt;0)
{
int max = arr[0];
for(i=0;i&lt;len;i++)
{
If(max=arr[i])
{
max = arr[i];
}
}
}
for(i=0;i&lt;len;i++)
arr[i]=max;
return arr;
}
```

Test case 1:
Input :

[2,5,8,11,3],5

Expected return value:

[11,11,11,11,11]

Test case 2:
Input :

[3,2,5,8,9,11,23,45,63],9

Expected return value:

[63,63,63,63,63,63,63,63,63]

Sample Input

```
5
1 4 5 3 2
```

Sample Output

```
5 5 5 5 5
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q6. The function descendingSort Array(int *arr,int len)accepts an integer array arr of length len(len≥0)as an input and performs an inplace sort operation on it.The functions is expected to return the input array sorted in descending order ,but instead ,it returns the array sorted in ascending order due to a bug in the code.

```
Int* descendingSortArray(int *arr, int len)

{
Int small,pos,i,j,temp;
for(i=0;i&lt;=len-1;i++)
{
for(j=i;j&lt;=len;j++)
{
temp=0;
if(arr[i]&lt;arr[j])
{
temp=arr[i];
arr[i]=arr[j];
arr[j]=temp;
}
}
}
return arr;
}
```

Testcasse 1:
Input :
[3,6,4,1,7,9,1,3,12,15],10
Expected return value:
[15,12,9,7,6,4,3,2,1,1]
Testcase 2:
Input:
[3,3,3,3,3,3,3,3,3],9
Expected return value:
[3,3,3,3,3,3,3,3,3]

Sample Input

```
5
1 5 7 4 6
```

Sample Output

```
7 6 5 4 1
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q7. 1. You are given a predefined class **Point** containing a collection of methods to perform some basic operations.

You will have to implement the function **isTriangle(Point p1,Point p2,Point p3)** which accepts 3 points as input and checks whether the given 3 points form the vertices of a triangle

If they form a triangle the function returns 1 else it returns 0

You are supposed to use **Point** structure and associated methods for the task.

PROGRAM:

```
public class Triangle
{
public static int isTriangle(point p1,point p2,point p3)
{
//write your code here
return 0;
}
}
```

Sample Input

```
4 5 6
```

Sample Output

```
Valid
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Answer Key & Solution

Section 1 - Automata Fix

Q1

Test Case

Input

```
january
26
```

Output

```
No
```

Weightage - 25

Input

```
july
16
```

Output

```
No
```

Weightage - 25

Input

```
march
5
```

Output

```
No
```

Weightage - 50

Sample Input

```
july
5
```

Sample Output

```
Yes
```

Solution

Header

```
#include<stdio.h>
#include<string.h>
int checkBirthday(char* month,int day)
{




//if(strcmp(month,"july")  ||   (day =5))
if(strcmp(month,"july")  == 0 &&  (day -5) == 0)
        return 1;
    else
        return 0;
```

Footer

```
}

int main()
{
    char inp[10];
```

```
    char inp[10];
    scanf("%s",inp);
    int day;
    scanf("%d",&day);
    if(checkBirthday(inp,day)==1)
        printf("Yes");
    else
        printf("No");
    return 0 ;
}
```

Q2

Test Case

Input

```
6
2 6 8 9 3 7
3
```

Output

```
2 6 8 9 7
```

Weightage - 50

Input

```
5
1 8 7 6 9
9
```

Output

```
1 8 7 6
```

Weightage - 50

Sample Input

```
5
1 2 3 4 5
3
```

Sample Output

```
1 2 4 5
```

Solution

Header

```
#include<stdio.h>
int deleteElement(int arr[], int n, int x)
{




if (arr[n-1] == x)
    return (n-1);
int prev = arr[n-1], i;
for (i=n-2; i>=0 && arr[i]!=x; i--)
{
    int curr = arr[i];
    arr[i] = prev;
    prev = curr;
}
if (i < 0)
    return 0;
arr[i] = prev;

return (n-1);
```

Footer

```
}
int main()
{
        int n;
        scanf("%d",&n);
    int arr[n];
        for(int i=0;i<n;i++){
                scanf("%d ",&arr[i]);
        }
    int x;
        scanf("%d",&x);
    n = deleteElement(arr, n, x);
    for (int i=0; i<n; i++)
    printf("%d ",arr[i] );

    return 0;
}
```

Q3

Test Case

Input

```
6
1 1 2 2 3 3
```

Output

```
1 0 1 0 1
```

Weightage - 50

Input

```
8
4 1 2 1 0 2 1 1
```

Output

```
0 0 0 0 0 0 1
```

Weightage - 50

Sample Input

```
4
1 1 2 2
```

Sample Output

```
1 0 1
```

Sample Input

```
5
1 1 1 1 2
```

Sample Output

```
1 1 1 0
```

Solution

Header

```
#include<stdio.h>
#include<stdlib.h>


void Manchester(int *arr, int len){
```

```
#include<stdio.h>
#include<stdlib.h>
void Manchester(int *arr, int len){
int* res = (int*)malloc(sizeof(int)*len);
for(int i= 1; i< len; i++){
 if(arr[i]==arr[i-1])
    res[i]=1;
 else
    res[i]=0;
}
 for(int i=1;i<len;i++)
        printf("%d ",res[i]);
}
int main()
{
    int arr[100],i,len;
    scanf("%d",&len);
    for(i=0;i<len;i++)
        scanf("%d",&arr[i]);
    Manchester(arr,len);
    return 0;
}
```

Footer

```
}
int main()
{
    int arr[100],i,len;
    scanf("%d",&len);
    for(i=0;i<len;i++)
        scanf("%d",&arr[i]);
    Manchester(arr,len);
    return 0;
}
```

Q4

Test Case

Input

```
8
1 2 3 4 5 6 7 8
```

Output

```
4.000000
```

Weightage - 50

Input

```
10
10 20 30 40 50 60 70 80 90 100
```

Output

```
50.000000
```

Weightage - 50

Sample Input

```
4
1 2 7 8
```

Sample Output

```
2.000000
```

Sample Input

```
6
10  20  100  45  201  459
```

Sample Output

```
45.000000
```

Solution

Header

```
#include<stdio.h>
void Array_sort(int *array , int n)
{
    int i=0 , j=0 , temp=0;
    for(i=0 ; i<n ; i++)
    {
        for(j=0 ; j<n-1 ; j++)
        {
            if(array[j]<array[j+1])
            {
                temp        = array[j];
                array[j]    = array[j+1];
                array[j+1]  = temp;
            }
        }
    }
}
float Find_median(int array[] , int n)
{
```

```
#include <stdio.h>
void Array_sort(int *array , int n)
{
    int i=0 , j=0 , temp=0;
    for(i=0 ; i<n ; i++)
    {
        for(j=0 ; j<n-1 ; j++)
        {
            if(array[j]<array[j+1])
            {
                temp        = array[j];
                array[j]    = array[j+1];
                array[j+1]  = temp;
            }
        }
    }
}
float Find_median(int array[] , int n)
{
    float median=0;
    if((n/2)!=1)
        median = array[n/2];
    return median;
}
int main()
{
    int array_1[30] = {0};
    int i=0 ,n=0;
    float median=0;
```

```
    scanf("%d",&n);
    for(i=0 ; i<n ; i++)
    {
        scanf("%d",&array_1[i]);
    }
    Array_sort(array_1 , n);
    median = Find_median(array_1 , n);
    printf("%f",median);
    return 0;
}
```

Footer

```
}
int main()
{
    int array_1[30] = {0};
    int i=0 ,n=0;
    float median=0;
    scanf("%d",&n);
    for(i=0 ; i<n ; i++)
    {
        scanf("%d",&array_1[i]);
    }
    Array_sort(array_1 , n);
    median = Find_median(array_1 , n);
    printf("%f",median);
    return 0;
}
```

Q5

Test Case

Input

```
5
1 8 4 6 9
```

Output

```
9 9 9 9 9
```

Weightage - 50

Input

```
7
34 5 6 3 2 7 9
```

Output

```
34 34 34 34 34 34 34
```

Weightage - 50

Sample Input

```
5
1 4 5 3 2
```

Sample Output

```
5 5 5 5 5
```

Solution

Header

```
#include<stdio.h>
int* maxReplace(int *arr, int len)
{



        int i, max;
        if(len>0)
        {
            //
          max=arr[0];
            for(i=0;i<len;i++)
            {
             //   int dummy;
                    if(max<arr[i])
                            max=arr[i];
            }
        //   dummy = 100;
        }
        for(i=0;i<len;i++)
            arr[i]=max;
            return arr;
```

Footer

```
}
int main()
{
int  size, ind;
scanf("%d",&size);
int arr[size];
for(int i=0;i<size;i++){
        scanf("%d",&arr[i]);
}
maxReplace(arr, size);
for(ind = 0; ind < size; ind++)
    printf("%d ", arr[ind]);

}
```

Q6

Test Case

Input

```
6
7 5 9 3 7 8
```

Output

```
9 8 7 7 5 3
```

Weightage - 50

Input

```
10
3 5 7 8 0 1 5 7 8 2
```

Output

```
8 8 7 7 5 5 3 2 1 0
```

Weightage - 50

Sample Input

```
5
1 5 7 4 6
```

Sample Output

```
7 6 5 4 1
```

Solution

Header

```
#include<stdio.h>
int * descendingSortArray(int *arr, int len)
{



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];
             arr[i] = arr[j];
             arr[j] = temp;
         }
    }
}
return arr;
```

Footer

```
}

int main()
{
int index, size;
scanf("%d",&size);
int arr[size];
for(int i=0;i<size;i++){
    scanf("%d",&arr[i]);
}
descendingSortArray(arr, size);
for(index = 0; index < size; index++)
   printf("%d " , arr[index]);
return 0;
}
```

Q7

Test Case

Input

```
1 9 26
```

Output

```
Invalid
```

Weightage - 50

Input

```
4 8 12
```

Output

```
Invalid
```

Weightage - 25

Input

```
6 7 8
```

Output

```
Valid
```

Weightage - 25

Sample Input

```
4 5 6
```

Sample Output

```
Valid
```

Solution

Header

```
#include<stdio.h>
int checkValidity(int a, int b, int c)
{



    if (a + b <= c || a + c <= b || b + c <= a)
        return 0;
    else
        return 1;

}
```

Footer

```
int main()
{
    int a, b, c;
    scanf("%d %d %d",&a,&b,&c);
    if (checkValidity(a, b, c))
        printf("Valid");
    else
        printf("Invalid");
}
```