10 (b) CTS Test 10

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. Below is a snippet to check whether a given array is a **Palindrome** or **Not Palindrome**. Complete palindrome(int arr[], int n) function to get the desired output.

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

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

Sample Input

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

Sample Output

```
Palindrome
```

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 return 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:

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

}
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;
}
```

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. The Function **arrayReverse(int *arr,int len)** accepts an array arr of length len(len >=0) as an argument.The function is expected to reverse the elements of the input array in-place.
For example, if the input array arr is {20,30,10,40,50} the function is expected to return{50,40,10,30,20}
You need to complete the function arrayReverse(int *arr, int len)

```
#include<stdio.h>
int* arrayReverse(int *arr,int len){
//write your code here
}
int main()
{
int n;
scanf("%d",&n);
int arr[n];
for(int i=0;i<n;i++){
   scanf("%d",&arr[i]);
}
arrayReverse(arr, n);
for(int i=0;i<n;i++){
   printf("%d ",arr[i]);
}
}
```

Sample Input

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

Sample Output

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

Time Limit: 2 ms Memory Limit: 256 kb Code Size: 256 kb

Q4. The function **patternPrint(int n)** supposed to print n numberof lines in the following pattern
For n=4 the pattern should be:

```
1
1 1
1 1 1
 1 1 1 1
```

Complete the function **patternPrint(int n)** to get the desired output

PROGRAM

```
#include<stdio.h>
void patternPrint(int num)
{
// write here
}
int main()
{
   int n;
   scanf("%d",&n);
   patternPrint(n);
}
```

Sample Input

```
5
```

Sample Output

```
1
11
111
1111
```

Time Limit: 2 ms Memory Limit: 256 kb Code Size: 256 kb

Q5. The function **matrixsum(int **matrix,int m,int n)** is supposed to return the sum of elements of the input array matrix having m rows and n columns. The logic is provided. But it is not giving the desired output due to a logical error. Find the logical error and fix it.

Program :

```
#include<stdio.h>
#define SIZE 100
int matrixsum(int row,int col)
{
   int sum=0;
   int arr[row][col];
   for(int i=0;i<row;i++){
   for(int j=0;j<col;j++)
   {
       scanf("%d",&arr[i][j]);
   }
   }
   for(int i=0;i<col;i++){
       for(int j=0;j<i;j++){
           sum=sum+arr[i][j];
       }
   }
   return sum;
}
int main()
{
   int m,n;
   scanf("%d %d",&m,&n);


   printf("%d",matrixsum(m,n));
}
```

Sample Input

```
3 3
1 2 3
4 5 6
7 8 9
```

Sample Output

```
45
```

Time Limit: 2 ms Memory Limit: 256 kb Code Size: 256 kb

Q6. You will have to implement the function **checkValidity(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

Find th elogical error that if found in the given snippet and correct it

PROGRAM:

```
#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;

}

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

Sample Input

```
4 5 6
```

Sample Output

```
Valid
```

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

Q7. The function **findMinElement(int *arr1,int len1,int *arr2,int len2)** accepts two integer arrays arr1,arr2 of length len1,len2 respectively.
It is supposed to return the smallest element in both the input arrays.
Another function **sortArray(int *arr,int len)** sorts the input array arr of length len in ascending order and returns the sorted array.
Your task is to use **sortArray(int *arr,int len)** function and find the error that is in the **findMinElement(int* arr1,int len1,int* arr2,int len2)** Function and fix it.

```
#include<stdio.h>
int * sortArray(int *arr, int length)
{
int x=0,y=0,n=length;
for(x=0;x<n;x++)
{
int index_of_min = x;
for(y=x;y<n;y++)
{
if(arr[index_of_min]>arr[y])
{
index_of_min=y;
}
}
int temp=arr[x];
arr[x]=arr[index_of_min];
arr[index_of_min]=temp;
}
return arr;
}
void minElement(int arr1[],int arr2[],int size){
 int index;
   sortArray(arr1, size);
   sortArray(arr2, size);
   int min=0;
   if(arr1[0]>arr2[0]){
       min=arr1[0];
   }
   else{
       min=arr2[0];
   }
  printf("%d",min);

}
int main()
{
   int size;
   scanf("%d",&size);
int arr1[size],arr2[size];
for(int i=0;i<size;i++){
   scanf("%d",&arr1[i]);
}
for(int i=0;i<size;i++){
   scanf("%d",&arr2[i]);
}
minElement(arr1,arr2,size);
return 0;
}
```

Sample Input

```
5
4  6  2  3  9
8  4  3  6  2
```

Sample Output

```
2
```

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

Answer Key & Solution

Section 1 - Automata Fix

Q1

Test Case

Input

```
10
1 2 3 4 5 4 3 2 2
```

Output

```
Not Palindrome
```

Weightage - 50

Input

```
8
1 2 3 4  4 3 2 1
```

Output

```
Palindrome
```

Weightage - 50

Sample Input

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

Sample Output

```
Palindrome
```

Solution

Header

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


    int flag = 0;
    for (int i = 0; i <= n / 2 && n != 0; i++) {
        if (arr[i] != arr[n - i - 1]) {
            flag = 1;
            break;
        }
    }
    if (flag == 1)
        printf("Not Palindrome");
    else
        printf("Palindrome");
```

Footer

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

```
    palindrome(arr, n);
    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

```
10
4 9 4 7 5 9 3 6 0 8
```

Output

```
8 0 6 3 9 5 7 4 9 4
```

Weightage - 25

Input

```
3
1 1 1
```

Output

```
1 1 1
```

Weightage - 25

Input

```
5
10 56 76 89 45
```

Output

```
45 89 76 56 10
```

Weightage - 25

Input

```
10
12 13 14 16 17 18 19 23 25 27
```

Output

```
27 25 23 19 18 17 16 14 13 12
```

Weightage - 25

Sample Input

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

Sample Output

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

Solution

Header

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



int i,temp,originallen=len;
for(i=0;i<originallen/2;i++){
temp=arr[len-1];
arr[len-1]=arr[i];
arr[i]=temp;
len-=1;
}
return arr;
```

Footer

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

Q4

Test Case

Input	Output
6	1 11 111 1111

Weightage - 50

Input	Output
20	1 11 111 1111

Weightage - 50

Sample Input	Sample Output
5	1 11 111 1111

Solution

Header

```
#include<stdio.h>
void patternPrint(int num)
{



    for(int i=0;i<num;i++){
        for(int j=0;j<=i;j++){
            printf("1");
        }
        printf("\n");
    }
```

Footer

```
}
int main()
{
    int n;
    scanf("%d",&n);
    patternPrint(n);
}
```

Q5

Test Case

Input

```
2
1 4
6 8
```

Output

```
10
```

Weightage - 25

Input

```
5 5
1 2 3 67 67
90 87 65 43 21
1 2 3 4 5
```

Output

```
511
```

Weightage - 75

Sample Input

```
3 3
1 2 3
4 5 6
7 8 9
```

Sample Output

```
45
```

Solution

Header

```
#include<stdio.h>
#define SIZE 100
int matrixsum(int row,int col)
{
```

```
    int sum=0;
    int arr[row][col];
    for(int i=0;i<row;i++){
    for(int j=0;j<col;j++)
    {
        scanf("%d",&arr[i][j]);
    }
    }
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            sum=sum+arr[i][j];
        }
    }
    return sum;
```

Footer

```
}
int main()
{
    int m,n;
    scanf("%d %d",&m,&n);


    printf("%d",matrixsum(m,n));
}
```

Q6

Test Case

Input	Output
1 9 26	Invalid

Weightage - 50

Input	Output
4 8 12	Invalid

Weightage - 25

Input	Output
6 7 8	Valid

Weightage - 25

Sample Input	Sample Output
4 5 6	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;

}

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

Q7

Test Case

Input

```
5
1 2 3 4 5
9 8 7 6 5
```

Output

```
1
```

Weightage - 50

Input

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

Output

```
3
```

Weightage - 50

Sample Input

```
5
4 6 2 3 9
8 4 3 6 2
```

Sample Output

```
2
```

Solution

Header

```
#include<stdio.h>
int * sortArray(int *arr, int length)
{
int x=0,y=0,n=length;
for(x=0;x<n;x++)
{
int index_of_min = x;
for(y=x;y<n;y++)
{
if(arr[index_of_min]>arr[y])
{
index_of_min=y;
}
}
int temp=arr[x];
arr[x]=arr[index_of_min];
arr[index_of_min]=temp;
}
return  arr;
}
void minElement(int arr1[],int arr2[],int size){
 int index;
 sortArray(arr1, size);
 sortArray(arr2, size);
 int min=0;
 if(arr1[0]>arr2[0]){
     min=arr2[0];
 }
 else{
     min=arr1[0];
 }
printf("%d",min);
```

Footer

```
}
int main()
{
    int size;
    scanf("%d",&size);
int arr1[size],arr2[size];
for(int i=0;i<size;i++){
    scanf("%d",&arr1[i]);
}
for(int i=0;i<size;i++){
    scanf("%d",&arr2[i]);
}
minElement(arr1,arr2,size);
return 0;
}
```