9 (b) CTS Test 9

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. swap all the odd bits into even bits and vice versa. Complete **swapBits(unsigned int x){}** to give the desired output.

```
#include <stdio.h>
unsigned int swapBits(unsigned int x)
{

}
int main()
{
    unsigned int x;
scanf("%u",&x);
    printf("%u ", swapBits(x));
    return 0;
}
```

Sample Input

```
23
```

Sample Output

```
43
```

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

Q2. 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 flag = 0;
    for (int i = 0; i <= n / 2 && n != 0; i++) {
        if (arr[i] != arr[n - 1]) {
            flag = 1;
            break;
        }
    }
    if (flag == 1)
        printf("Not Palindrome");
    else
        printf("Palindrome");
}
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

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

```
int arrayReverse(int *arr,int len){
int i,temp,originallen=len;
for(i=0;i<originallen;i++){
temp=arr[len-1];
arr[len-1]=arr[i];
arr[i]=temp;
len-=1;
}
Return arr;
}
```

Input Format

First Line of input contains the integer **n** - size of the array
Second line contains n - space seperated integers

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/method **countElement(int arr[],int size,int num)** is supposed to return the number of elements in the input array arr which are greater than twice the input number **num.** complete the function/method **countElement(int arr[],int size,int num)** to give the desired output.

Input Format

First line of input consist of integer **size** - size of the array arr
Second line contains size number of space seperated integers
Third line contains the integer **num**

Sample Input

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

Sample Output

```
1
```

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

Q5. You are given a pre-defined structure Point and also a collection of related functions which can be used to perform some basic operations on the structure.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 can make a triangle or not. If they make a triangle the function returns 1 else if returns 0

Sample Input

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

Sample Output

```
Yes
```

Sample Input

```
-6 -9
-6 -9
-3 -5
```

Sample Output

```
No
```

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

Q6. **QUESTION**

The function multiplynumber(int a , int b,int c)accepts three integers a,b and c as inputs,it is supposed to return the multiplication productive of the maximum two or three input numbers

```
#include<stdio.h>
#define max(a,b) (a)>(b)?(a):(b)
#define min(a,b) (a)<(b)?(a):(b)
int multiplyNumber(int a, int b, int c)
{
int result, min, max, mid;
max=(Max(a,b),c);
min=(Min(a,b),c);
mid=(a+b+c)-(min-max);
result=(max*mid);
return result;
}
int main()
{
  int a,b,c,ans;
  scanf("%d%d%d",&a,&b,&c);
  ans=multiplyNumber(a,b,c);
  printf("%d",ans);
  return 0;
}
```

Sample Input

```
1 48 7
```

Sample Output

```
336
```

Sample Input

```
12 15 98
```

Sample Output

```
1470
```

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

Q7. **Find the factorial of a given number.**

Sample Input

```
5
```

Sample Output

```
120
```

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

Answer Key & Solution

Section 1 - Automata Fix

Q1

Test Case

Input

```
46
```

Output

```
29
```

Weightage - 50

Input

```
15648
```

Output

```
15888
```

Weightage - 50

Sample Input

```
23
```

Sample Output

```
43
```

Solution

Header

```
#include <stdio.h>
unsigned int swapBits(unsigned int x)
{
```

```
    unsigned int even_bits = x & 0xAAAAAAAA;
    unsigned int odd_bits = x & 0x55555555;
    even_bits >>= 1;
    odd_bits <<= 1;
    return (even_bits | odd_bits);
```

Footer

```
}
int main()
{
    unsigned int x;
    scanf("%u",&x);
    printf("%u ", swapBits(x));
    return 0;
}
```

Q2

Test Case

Input

Output

10 1 2 3 4 5 4 3 2 2	Not Palindrome

Weightage - 50

Input	Output
8 1 2 3 4 4 3 2 1	Palindrome

Weightage - 50

Sample Input	Sample Output
5 1 2 3 2 1	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;
}
```

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

```
10
11 12 13 14 15 16 17 18 19 20
8
```

Output

```
4
```

Weightage - 50

Input

```
6
12 12 13 14 23
6
```

Output

```
6
```

Weightage - 50

Sample Input

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

Sample Output

```
1
```

Solution

Header

```
#include<stdio.h>
int countElement(int arr[],int size,int num)
{



 int count=0;
    for(int i=0;i<size;i++)
    {
        if(arr[i]>2*num)
        {
            count++;
        }
    }
    return count;
```

Footer

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

Q5

Test Case

Input

```
1 3
3 5
6 5
```

Output

```
Yes
```

Weightage - 25

Input

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

Output

```
Yes
```

Weightage - 25

Input

```
10 5
-10 -5
4 5
```

Output

```
Yes
```

Weightage - 25

Input

```
-1 -3
-6 -5
-6 -5
```

Output

```
No
```

Weightage - 25

Sample Input

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

Sample Output

```
Yes
```

Sample Input

```
-6 -9
```

Sample Output

```
No
```

```
-6  -9
-3  -5
```

```
NO
```

Solution

Header

```
#include<stdio.h>
#include<math.h>
#include<malloc.h>
typedef struct POINT
{
    int x,y;
}Point;
int getX(Point *p)
{
    return p -> x;
}
int getY(Point *p)
{
    return p -> y;
}
void setPoint( Point *p , int x , int y)
{
    p->x = x;
    p->y = y;
}
double findDist( Point *p1 , Point *p2)
{
     return sqrt( (p1->x -p2->x) * (p1->x - p2->x) + (p1->y -p2->y) * (p1->y -p2->y) );

}



int isTriangle( Point *p1 , Point * p2 , Point *p3)
{
    double a , b, c;
    a = findDist(p1,p2);//sqrt( (p1->x -p2->x) * (p1->x - p2->x) + (p1->y -p2->y) * (p1->y -p2->y) );
    b = findDist(p2,p3);//sqrt( (p2->x -p3->x) * (p2->x - p3->x) + (p2->y -p3->y) * (p2->y -p3->y) );
    c = findDist(p1,p3);//sqrt( (p1->x -p3->x) * (p1->x - p3->x) + (p1->y -p3->y) * (p1->y -p3->y) );

    if( ((a+b) > c) && ((a+c) > b) && ((b+c) > a))return 1;
    else return 0;
}
```

Footer

```
int main()
{
    int x,y;
    Point *p1 , *p2,*p3;
    scanf("%d %d",&x,&y);
    p1 = (Point*)malloc(sizeof(Point));
    setPoint(p1,x,y);
    scanf("%d %d",&x,&y);
        p2 = (Point*)malloc(sizeof(Point));

    setPoint(p2,x,y);
    scanf("%d %d",&x,&y);
        p3 = (Point*)malloc(sizeof(Point));
```

```
    setPoint(p3,x,y);
    if( isTriangle(p1,p2,p3) == 1 )
       printf("Yes");
    else
       printf("No");
       return 0;
}
```

Q6

Test Case

Input

```
1  2  3
```

Output

```
6
```

Weightage - 50

Input

```
78  12  100
```

Output

```
7800
```

Weightage - 50

Sample Input

```
1  48  7
```

Sample Output

```
336
```

Sample Input

```
12  15  98
```

Sample Output

```
1470
```

Solution

Header

```
#include<stdio.h>
#define Max(a,b) (a)>(b)?(a):(b)
#define Min(a,b) (a)<(b)?(a):(b)
int multiplyNumber(int a, int b, int c)
{
```

```
#include<stdio.h>
#define Max(a,b) (a)>(b)?(a):(b)
#define Min(a,b) (a)<(b)?(a):(b)
int multiplyNumber(int a, int b, int c)
{
int result, min, max, mid;
max=Max(Max(a,b),c);
min=Min(Min(a,b),c);
mid=(a+b+c)-(min+max);
result=(max*mid);

return result;
```

```
}
int main()
{
    int a,b,c,ans;
    scanf("%d%d%d",&a,&b,&c);
    ans=multiplyNumber(a,b,c);
    printf("%d",ans);
    return 0;
}
```

Footer

```
int main()
{
    int a,b,c,ans;
    scanf("%d%d%d",&a,&b,&c);
    ans=multiplyNumber(a,b,c);
    printf("%d",ans);
    return 0;
}
```

Q7

Test Case

Input	Output
10	3628800

Weightage - 50

Input	Output
4	24

Weightage - 50

Sample Input	Sample Output
5	120

Solution

Header

```
#include<stdio.h>
int factorial(int n)
{
```

```
#include<stdio.h>
int factorial(int n)
{
```

```
    int itr,fact=1;
for(itr =1; itr <= n; itr++)
{
fact = fact * itr;
}
return fact;
}
int main()
{
long int fact, n, i;
scanf("%d", &n);
fact=factorial(n);
printf("%d", fact);
return 0;
}
```

Footer

```
}
int main()
{
int fact, n, i;
scanf("%d", &n);
fact=factorial(n);
printf("%d", fact);
return 0;
}
```