4 (b) CTS Test 4

Test Summary

- No. of Sections: 1
- No. of Questions: 7
- Total Duration: 20 min

Section 1 - Automata Fix

Section Summary

- No. of Questions: 7
- Duration: 20 min

Additional Instructions:

None

Q1. Find the sum of first n natural numbers.

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

}
int main()
{
int n;
scanf("%d",&n);
printf("%d",findSum(n));
return 0;
}
```

Sample Input

```
5
```

Sample Output

```
15
```

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

Q2. Given two integers A and B. The task is to count how many numbers in the interval [A, B] have an odd number of divisors.

Examples:

Input : A = 1, B = 10
Output : 3

Input : A = 5, B = 15
Output : 1

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

}
int main()
{
    int a, b;
printf("%d",OddDivCount(a,b));
    return 0;
}
```

Sample Input

```
10 20
```

Sample Output

```
1
```

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

Q3. Write the remaining piece of code to find the nth fibinocci series

```
#include <stdio.h>
int fib(int n)
{
}
int main()
{
int n ;
scanf("%d",&n);
printf("%d", fib(n));
getchar();
return 0;
}
```

Sample Input

```
6
```

Sample Output

```
8
```

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

Q4. Find the logical error in the below code.

```
void main () {
int i, j, n = 5;
for(i=1; i<n; i++)
{
for(j=i;j<n;j++);
{
printf("%d", i);
}
printf("\n");
}
}
```

Sample Input

```
```

Sample Output

```
1111
222
33
4
```

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

Q5. Find the gcd of two numbers. The entire code is given below with logical errors. Correct it.

```
// C program to find GCD of two numbers
#include <stdio.h>
// Recursive function to return gcd of a and b
int gcd(int a, int b)
{

}
// Driver program to test above function
int main()
{
   int a,b;
scanf("%d %d",&a,&b);
   printf("GCD of %d and %d is %d ", a, b, gcd(a, b));
   return 0;
}
```

Sample Input

```
10 5
```

Sample Output

```
GCD of 10 and 5 is 5
```

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

Q6. calculate the nth amstrong number.
Fill the logic in the provided function

Sample Input

```
153
```

Sample Output

```
True
```

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

Q7. The function **findMaxElement(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 largest 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 complete the code in **findMaxElement(int arr1[],int len1,int arr2[],int len2)** so that it passes all test cases.

Sample Input

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

Sample Output

```
9
```

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

Answer Key & Solution

Section 1 - Automata Fix

Q1 **Test Case**

Input	Output
20	210

Weightage - 50

Input	Output
14	105

Weightage - 50

Sample Input	Sample Output
5	15

Solution

Header

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

```
#include<stdio.h>
int findSum(int n)
{
int sum = 0;
for (int x=1; x<=n; x++)
    sum = sum + x;
return sum;
}
int main()
{
int n;
scanf("%d",&n);
printf("%d",findSum(n));
return 0;
}
```

Footer

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

```
printf("%d",findSum(n));
return 0;
}
```

Q2

Test Case

Input

```
20 50
```

Output

```
3
```

Weightage - 50

Input

```
33 77
```

Output

```
3
```

Weightage - 50

Sample Input

```
10 20
```

Sample Output

```
1
```

Solution

Header

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



// Function to count numbers having odd
// number of divisors in range [A, B]
#include<stdio.h>
int OddDivCount(int a, int b)
{
    // variable to odd divisor count
    int res = 0;
    // iterate from a to b and count their
    // number of divisors
    for (int i = a; i <= b; ++i) {

        // variable to divisor count
        int divCount = 0;
        for (int j = 1; j <= i; ++j) {
            if (i % j == 0) {
                ++divCount;
            }
        }

        // if count of divisor is odd
        // then increase res by 1
        if (divCount % 2) {
            ++res;
        }
    }
    return res;
}

// Driver code
int main()
{
    int a, b;
    scanf("%d%d",&a,&b);
    printf("%d",OddDivCount(a,b));
    return 0;
}
```

Footer

```
}
int main()
{
    int a, b;
    printf("%d",OddDivCount(a,b));
    return 0;
}
```

Q3

Test Case

Input	**Output**
10	55

Weightage - 40

Input	**Output**
18	2584

Weightage - 60

Sample Input	**Sample Output**
6	8

Solution

Header

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

```
#include <stdio.h>
int fib(int n)
{
    if (n <= 1)
        return n;
    return fib(n - 1) + fib(n - 2);
}

int main()
{
    int n;
    scanf("%d",&n);
    printf("%d", fib(n));
    getchar();
    return 0;
}
```

Footer

```
}

int main()
{
    int n;
    scanf("%d",&n);
    printf("%d", fib(n));
    getchar();
    return 0;
}
```

Q4

Test Case

Input

Output

```
1111
222
33
```

Weightage - 100

Sample Input

Sample Output

```
1111
222
33
```

Solution

Header

```
#include<stdio.h>
int main () {
  int i, j, n = 5;
```

```
#include<stdio.h>
int main(){
  int i, j, n = 5;
  for(i=1; i<n; i++)
  {
              for(j=i;j<n;j++)
{
printf("%d", i);
   }
    printf("\n");
  }
}
```

Footer

```
  }
```

Q5 **Test Case**

Input

```
20 4
```

Output

```
GCD of 20 and 4 is 4
```

Weightage - 50

Input

```
45 9
```

Output

```
GCD of 45 and 9 is 9
```

Weightage - 50

Sample Input

```
10 5
```

Sample Output

```
GCD of 10 and 5 is 5
```

Solution

Header

```
#include <stdio.h>
// Recursive function to return gcd of a and b
int gcd(int a, int b)
{
```

```
#include <stdio.h>
// Recursive function to return gcd of a and b
int gcd(int a, int b)
{
    if (b == 0)
        return a;

    return gcd(b, a % b);
}
// Driver program to test above function
int main()
{
    int a,b;
    scanf("%d %d",&a,&b);
    printf("GCD of %d and %d is %d ", a, b, gcd(a, b));
    return 0;
}
```

Footer

```
}
int main()
{
    int a,b;
    scanf("%d %d",&a,&b);
    printf("GCD of %d and %d is %d ", a, b, gcd(a, b));
    return 0;
}
```

Q6

Test Case

Input	Output
156	False

Weightage - 50

Input	Output
121	False

Weightage - 25

Input	Output
153	True

Weightage - 25

Sample Input	Sample Output
153	True

Solution

Header

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

```
int isArmstrong(int x)
{




// C program to find Armstrong number

#include <stdio.h>

/* Function to calculate x raised to the power y */
int power(int x, unsigned int y)
{
    if (y == 0)
        return 1;
    if (y % 2 == 0)
        return power(x, y / 2) * power(x, y / 2);
    return x * power(x, y / 2) * power(x, y / 2);
}

/* Function to calculate order of the number */
int order(int x)
{
    int n = 0;
    while (x) {
        n++;
        x = x / 10;
    }
    return n;
}

// Function to check whether the given number is
// Armstrong number or not
int isArmstrong(int x)
{
    // Calling order function
    int n = order(x);
    int temp = x, sum = 0;
    while (temp) {
        int r = temp % 10;
        sum += power(r, n);
        temp = temp / 10;
    }

    // If satisfies Armstrong condition
    if (sum == x)
        return 1;
    else
        return 0;
}

// Driver Program
int main()
{
    int x;
    scanf("%d",&x);
    if (isArmstrong(x) == 1)
        printf("True\n");
    else
        printf("False\n");

    return 0;
}
```

Footer

```
}

int main()
{
    int x;
    scanf("%d",&x);
    if (isArmstrong(x) == 1)
        printf("True\n");
    else
        printf("False\n");

    return 0;
}
```

Q7 **Test Case**

Input

```
10 6
1 8 5 7 2 4 12 8 9 11
3 5 7 12 98 89
```

Output

```
98
```

Weightage - 25

Input

```
5 5
1 2 3 4 5
5 4 3 2 6
```

Output

```
6
```

Weightage - 25

Input

```
5 9
8 99 89 90 11
78 999 888 77 666 545 67 54 23
```

Output

```
999
```

Weightage - 50

Sample Input

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

Sample Output

```
9
```

Solution

Header

```
#include<stdio.h>
#include<stdlib.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 findMaxElement(int arr1[],int arr2[],int len1,int len2){



    int index;
    sortArray(arr1,len1);
    sortArray(arr2, len2);
    int max=0;
    if(arr1[len1-1]>arr2[len2-1]){
        max=arr1[len1-1];
    }
    else{
        max=arr2[len2-1];
    }
   printf("%d",max);
```

Footer

```
}
int main()
{
    int len1,len2;
    scanf("%d %d",&len1,&len2);
    int arr1[len1],arr2[len2];
    for(int i=0;i<len1;i++){
        scanf("%d",&arr1[i]);
    }
    for(int i=0;i<len2;i++){
        scanf("%d",&arr2[i]);
    }
    findMaxElement(arr1,arr2,len1,len2);
    return 0;
}
```