8 (b) CTS Test 8

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. **PrintPrime**

You will have to implement the **methods PrintPrime(int num,int n**)which accepts initial number num and number of digits n as inputs and prints all the numbers of n digits starting from the initial number num

Sample Input

```
50 3
```

Sample Output

```
53,59,61,67,71,73,79,83,89,97,101,103,107,109,113,127,
```

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

Q2. **Compete Cell**

A colony of eight houses, represented as cells, are arranged in a straight line. Each day every cell compete with its adjacent cells(neighbours). An integer value of 1 represents an active cell and value of 0 represents an inactive cell.If both the cell are active or inactive , the cell becomes inactive the next day, otherwise it becomes active on the next day. The two cells on the ends have single adjacent cell, so the other adjacent cell can be assumed to be always inactive. Even after updating the cell state. consider its previous state for updating the state of other cells. Update the cell information of all cells simultaneously. Write a function cellCompete which takes takes one 8 element array of integers cells representing the current state of 8 cells and one integer days representing the number of days to simulate. An integer value of 1 represents an active cell and value of 0 represents an inactive cell.

```
Int* cellcompete(int* cells, int days)
{
}
```

Write an algorithm to output the state of the cells after given number of days

Input Format

The input to the function/ method consist of two arguments
states , a list of integers representing the current state of cells
days , an integer representing the number of days

Output Format

Return the list of integers representing the state of the cells after the given number of days

Sample Input

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

Sample Output

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

Sample Input

```
0 1 0 1 0 1 1 0
3
```

Sample Output

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

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

Q3. You are given a pre-defined structure Point and also a collection of related functions which can be used toperform some basi
operations on the structure.You will have to implement the function isRightTriangle(Point *P1, Point *P2,Point *P3) which acc

points as input and checks whether the given 3 points can make a right angle triangle or not. If they make a right angle triangle the function returns 1 else if returns 0

Sample Input

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

Sample Output

```
Yes
```

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

Q4.

Convert Binary to Decimal by using the existing function. Find the logical error given in the code.

```
void binarytodecimal(number)
{
 int dval=0, base=1, rem;
 while(number > 1)
 {
       rem = number % 10;
       dval = dval + rem * base;
         number = number / 10;
         base = base * 2;
 }
 return dval;
}
void main()
{
int num;
scanf(“%d”, &num);
printf(“%d”, binarytodecimal(num);
}
```

Sample Input

```
1011
```

Sample Output

```
11
```

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

Q5. 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 sum of largest element in each of 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
1 4 7 9 3
6 3 7 9 2
```

Sample Output

```
18
```

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

Q6. The function **pyramid(int n)** is a function that will print a reverse character pyramid pattern like the below one. Find the logical error to get the desired output

Input:

5

Output:

ABCDEFGHIJ
ABCDEFGH
ABCDEF
ABCD
AB

```
#include<stdio.h>
void pyramid(int n)
{
   int i, j, num, gap;
```

```
    for (i = n; i >= 1; i--) {
        for (gap = n - 1; gap >= 1; gap--) {
            printf(" ");
            printf(" ");
        }
        num = 'A';
        for (j = 1; j <= i; j++) {
printf("%c",num++);
        }
        for (j = i - 1; j >= 0; j--) {
            printf("%c",num++);
        }
        printf("\n");
    }
}
int main()
{
    int n;
scanf("%d",&n);
    pyramid(n);
    return 0;
}
```

Sample Input

```
5
```

Sample Output

```
ABCDEFGHIJ
  ABCDEFGH
    ABCDEF
      ABCD
```

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

Q7. The function **matrixsum(int **matrix,int m,int n)** is supposed to return the sum of diagonal elements of the input array matrix having m rows and n columns. Find the logical error in the function matrixsum and rectify it.

```
#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;
}
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

```
15
```

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

Answer Key & Solution

Section 1 - Automata Fix

Q1

Test Case

Input

```
50 2
```

Output

```
53,59,61,67,71,73,79,83,89,97
```

Weightage - 25

Input

```
100 3
```

Output

```
101,103,107,109,113,127,131,137,139,149,151,157,163,
```

Weightage - 25

Input

```
10 2
```

Output

```
11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,7
```

Weightage - 25

Input

```
1000 4
```

Output

```
1009,1013,1019,1021,1031,1033,1039,1049,1051,1061,10
```

Weightage - 25

Sample Input

```
50 3
```

Sample Output

```
53,59,61,67,71,73,79,83,89,97,101,103,107,109,113,12
```

Solution

Header

```
#include<stdio.h>
#include<math.h>
int isPrime(int num)
{
int factor, till;
if(num == 1) return 0;
if(num == 2) return 1;
till = sqrt(num);
for(factor = 2; factor <=till; factor++)
if(num % factor ==0 ) return 0;
return 1;
}
```

```
int power(int digit)
{
int power = 1, ctr;
for(ctr =1; ctr <= digit; ctr++)
power*=10;
return power;
}



int printPrime(int start, int nod)
// 373, 2
{
int from , to, num, safe=-1;
from = start;
to = power(nod);
if(from >  to) printf("INVALID INPUT");
else
for(num = from; num < to; num++)
{
    if(isPrime(num) == 1)
    {
        if(safe != -1)
             printf("%d,", safe);
             safe = num;
  }
}
if(safe!= -1)
    printf("%d", safe);
return 0;
}
```

Footer

```
int main()
{
    int num,digit;
    scanf("%d %d",&num,&digit);
printPrime(num, digit);
return 0;
}
```

Q2

Test Case

Input

```
1 1 1 0 1 1 1 1
2
```

Output

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

Weightage - 20

Input

```
0 0 0 0 1 1 1 1
5
```

Output

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

Weightage - 20

Input

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

Output

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

Weightage - 20

Input

```
1 1 1 1 0 1 1 1
2
```

Output

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

Weightage - 20

Input

```
1 1 0 0 0 0 1 0
10
```

Output

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

Weightage - 20

Sample Input

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

Sample Output

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

Sample Input

```
0 1 0 1 0 1 1 0
3
```

Sample Output

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

Solution

Header

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



int* cellCompete( int* arr  , int days)
{
    int ctr ,prev , nextprev ,size=8;
while( days)
{
    prev = 0;
    for( ctr = 0 ; ctr < size-1 ; ctr++)
    {
        nextprev = arr[ctr];
        arr[ctr] = prev ^ arr[ctr+1];
        prev = nextprev;
    }
    arr[ctr] = prev ^ 0;
    days--:
```

```
    }
    return arr;
    }
```

Footer

```
int main()
{
int *arr, size = 8 , days, index,ctr;
arr = (int*)malloc(sizeof(int)*8);
for(index=0 ; index<8 ; index++)
    scanf("%d",&arr[index]);
scanf("%d",&days);
arr=cellCompete( arr , days);
for( ctr = 0 ; ctr < size ; ctr++)
        printf("%d " , arr[ctr]);
return 0;
}
```

Q3

Test Case

Input

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

Output

```
No
```

Weightage - 25

Input

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

Output

```
Yes
```

Weightage - 25

Input

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

Output

```
No
```

Weightage - 25

Input

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

Output

```
No
```

Weightage - 25

Sample Input

Sample Output

3 4 3 1 8 4	Yes

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;
}



int isRightTriangle( Point *p1 , Point * p2 , Point *p3)
{
    double ab , bc, ac,a1,a2,a3;
    ab = sqrt( (p1->x -p2->x) * (p1->x - p2->x) + (p1->y -p2->y) * (p1->y -p2->y) );
    bc = sqrt( (p2->x -p3->x) * (p2->x - p3->x) + (p2->y -p3->y) * (p2->y -p3->y) );
    ac = sqrt( (p1->x -p3->x) * (p1->x - p3->x) + (p1->y -p3->y) * (p1->y -p3->y) );
    a1 = sqrt( ab * ab + ac * ac);
    a2 = sqrt( ab * ab + bc * bc);
    a3 = sqrt( bc * bc + ac * ac);

    if( a1 == bc ) return 1;
    else if( a2 == ac )return 1;
    else if( a3 == ab) 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( isRightTriangle(p1,p2,p3) == 1 )
           printf("Yes");
        else
           printf("No");
           return 0;
    }
```

Q4

Test Case

Input

```
1111
```

Output

```
15
```

Weightage - 25

Input

```
1110
```

Output

```
14
```

Weightage - 25

Input

```
111111101
```

Output

```
509
```

Weightage - 50

Sample Input

```
1011
```

Sample Output

```
11
```

Solution

Header

```
#include<stdio.h>
int binarytodecimal(int number)
{



#include<stdio.h>
int binarytodecimal(int number)
{
  int dval=0, base=1, rem;
  while(number > 0)
  {
              rem = number % 10;
              dval = dval + rem * base;
                  number = number / 10;
```

```
                    base = base * 2;
    }
    return dval;
}
int main(){
    int n;
    scanf("%d",&n);
    int ans = binarytodecimal(n);
    printf("%d",ans);
    return 0;
}
```

Footer

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

Q5

Test Case

Input

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

Output

```
15
```

Weightage - 50

Input

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

Output

```
11
```

Weightage - 50

Sample Input

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

Sample Output

```
18
```

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 maxElement(int arr1[],int arr2[],int size){




    int index;
    sortArray(arr1, size);
    sortArray(arr2, size);
    int max=0;

  printf("%d",arr1[size-1]+arr2[size-1]);
```

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]);
  }
  maxElement(arr1,arr2,size);
  return 0;
  }
```

Q6

Test Case

Input	Output
6	ABCDEFGHIJKL ABCDEFGHIJ ABCDEFGH ABCDEF

Weightage - 50

Input	Output
9	ABCDEFGHIJKLMNOPQR ABCDEFGHIJKLMNOP ABCDEFGHIJKLMN ABCDEFGHIJKL

Weightage - 50

Sample Input

```
5
```

Sample Output

```
ABCDEFGHIJ
  ABCDEFGH
    ABCDEF
      ABCD
```

Solution

Header

```
#include<stdio.h>
void pyramid(int n)
{
    int i, j, num, gap;
    for (i = n; i >= 1; i--) {
        for (gap = n - 1; gap >= i; gap--) {
            printf(" ");
            printf(" ");
        }
        num = 'A';
        for (j = 1; j <= i; j++) {
            printf("%c",num++);
        }
        for (j = i - 1; j >= 0; j--) {
             printf("%c",num++);
        }
        printf("\n");
    }
```

Footer

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

Q7

Test Case

Input

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

Output

```
12
```

Weightage - 100

Sample Input

Sample Output

3 3 1 2 3 4 5 6 7 8 9	15

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++){
            if(i==j)
            sum=sum+arr[i][j];
        }
    }
    return sum;
```

Footer

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

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