TCS DIGITAL – CODING TEST

QUESTION 1

Problem Statement-: Given a square matrix, calculate the absolute difference between the sums of its diagonals.

1	2	3
4	5	6
9	8	9

The left-to-right diagonal = 1+5+9 = 15; The right to left diagonal = 3+5+9 = 17; Their absolute difference is |15-17| = 2

Function Description: diagonalDifference takes the following parameter: int arr[n][m]*:* an array of integers

Input Format: The first line contains a single integer, n the number of rows and columns in the square matrix arr Each of the next n lines describes a row, arr[i] and consists of n space-separated integers arr[i][j]

Output Format: Return the absolute difference between the sums of the matrix's two diagonals as a single integer.

Sample Input:

```
3
11 2 4
4 5 6
10 8 -12
```

Sample Output:

```
15
```

PROGRAMS:

Python

```
def diagonalDifference(arr):
        sum1 = 0
        sum2 = 0
        for i in range(len(arr)):
        sum1+=arr[i][i]
        sum2+=arr[i][n-i-1]
        return abs(sum1-sum2)
n = int(input())
arr = []
for _ in range(n):
        arr.append(list(map(int, input().rstrip().split())))

result = diagonalDifference(arr)
```

C

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int n, i, j, sum1 = 0, sum2 = 0, result;
    scanf("%d",&n);
    int array[n][n];
    for(i=0; i<n; i++)
    {
       for(j=0; j<n; j++)
        {
            scanf("%d",&array[i][j]);
        }
    }
    for(i=0; i<n; i++)
    {
       sum1 = sum1 + array[i][i];
        sum2 = sum2 + array[i][n-i-1];
    }
    result = abs(sum1 - sum2);
    printf("%d",result);
    return 0;
}
```

QUESTION 2

Problem Statement:- Jaya invented a Time Machine and wants to test it by time-traveling to visit Russia on the Day of Programmer (the 256thday of the year) during a year in the inclusive range from 1700 to 2700. From 1700 to 1917 , Russia's official calendar was the Julian Calendar since 1919 they used the Gregorian calendar system. The transition from the Julian to Gregorian calendar system occurred in 1918 , when the next day after 31 January was February 14 . This means that in 1918, February 14 was the 32nd day of the year in Russia. In both calendar systems, February is the only month with a variable amount of days; it has 29 days during a *leap year*, and 28 days during all other years. In the Julian calendar, leap years are divisible by 4 ; in the Gregorian calendar, leap years are either of the following:

- Divisible by 400
- Divisible by 4 and not divisible by 100

Given a year, y, find the date of the 256th day of that year *according to the official Russian calendar during that year*. Then print it in the format dd.mm.yyyy, where dd is the two-digit day, mm is the two-digit month, and yyyy is y.

For example, the given year is 1984.1984 is divisible by 4, so it is a leap year. The 256 day of a leap year after 1918 is September 12, so the answer is 12.9.1984.

Function Description

- Complete the *programmerday* function in the editor below. It should return a string representing the date of the 256th day of the year given.
- programmerday has the following parameter(s):
  - year: an integer

Input Format: A single integer denoting year y.

Output Format: Print the full date of programmerday during year y in the format dd.mm.yyyy, where dd is the two-digit day, mm is the two-digit month, and yyyy is y.

Sample Input: 2017

Sample Output: 13.09.2017

PROGRAMS

Python

```
def dayOfProgrammer(year):
    if year>=1700 and year<=1917:
        if year%4==0:
            return "12.09."+str(year)
        else:
            return "13.09."+str(year)
    elif year==1918:
        return "26.09."+str(year)
    else:
        a=not(year%100)
        if (( year%400 == 0)or (( year%4 == 0 ) and ( year%100 != 0))):
            return "12.09."+str(year)
        else:
            return "13.09."+str(year)
year = int(input().strip())
result = dayOfProgrammer(year)
print(result)
```

Output

```
2017
13.09.2017
```

C

```
#include<stdio.h>
int main()
{
    int year;
    scanf("%d",&year);
    if(year>=1700 && year<=1917)
    {
       if(year%4==0)
       {
           printf("12.09.%d",year);
       }
       else
       {
           printf("13.09.%d",year);
       }
    }
    else if(year==1918)
    {
        printf("26.09.%d",year);
    }
    else
    {
        if((year%400==0)||((year%4==0)&&(year%100!=0)))
        {
            printf("12.09.%d",year);
        }
        else
        {
            printf("13.09.%d",year);
        }
    }
   return 0;
}
```

QUESTION 3

Problem Statement:- Hobo's Drawing teacher asks his class to open their books to a page number. Hobo can either start turning pages from the front of the book or from the back of the book. He always turns pages one at a time. When she opens the book, page 1 is always on the right side: When he flips page 1, he sees pages 2 and 3. Each page except the last page will always be printed on both sides. The last page may only be printed on the front, given the length of the book.

If the book is n pages long, and he wants to turn to page p, what is the minimum number of pages he will turn?

He can start at the beginning or the end of the book.

Given n and p, find and print the minimum number of pages Hobo must turn in order to arrive at page p

Function Description: Complete the *countpage* function in the editor below.

It should return the minimum number of pages Hobo must turn.

countpage has the following parameter(s):

- *n*: the number of pages in the book
- *p*: the page number to turn to

Input Format

- The first line contains an integer n, the number of pages in the book.
- The second line contains an integer, p, the page that Hobo's teacher wants her to turn to.

Output Format: Print an integer denoting the minimum number of pages Hobo must turn to get to page p

Sample Input

```
6
2
```

Sample Output: 1

PROGRAMS:

Python

```
def pageCount(n, p):
   return min(p//2, n//2 - p//2)
n = int(input())
p = int(input())
result = pageCount(n, p)
print(result)
```

Output

```
8
4
2
```

C

```
#include <stdio.h>
int main()
{
   int n, p, min;
   scanf("%d",&n);
```

```
    scanf("%d",&p);
    min = (n/2)-(p/2);
    if(min>p/2)
    {
        min = p/2;
    }
    printf("%d",min);
    return 0;
}
```

QUESTION 4

Problem Statement:- Chacha is playing an arcade game and wants to climb to the top of the leader board and wants to track her ranking. The game uses Dense Ranking so its leader board works like this:

- The player with the highest score is ranked number 1 on the leader board.
- Players who have equal scores receive the same ranking number, and the next player(s) receive the immediately following ranking number.

For example, the four players on the leader board have high scores of 100, 90, 90, and 80. Those players will have ranks 1, 2, 2, and 3, respectively. If Chacha's scores are 70, 80 and 105, her rankings after each game are 4th, 3rd and 1st**.**

Function Description

Write a function *climbing*. It should return an integer array where each element res[j] represents Chacha's rank after the j game.

climbing has the following parameter(s):

- *scores*: an array of integers that represent leaderboard scores
- *alice*: an array of integers that represent Chacha's scores

Input Format

- The first line contains an integer n, the number of players on the leaderboard.
- The next line contains n space-separated integers scores[i], the leaderboard scores in decreasing order.
- The next line contains an integer, m, denoting the number games Chacha plays.
- The last line contains m space-separated integers chacha[j], the game scores.

Sample Input

```
7
100 100 50 40 40 20 10
4
5 25 50 120
```

Sample Output

```
6
4
2
1
```

PROGRAMS:

Python

```
def climbing(scores, chacha):
    unique_scores = list(reversed(sorted(set(scores))))

    i = len(chacha)-1
    j = 0
    ans = []

    while i >= 0:
        if j >= len(unique_scores) or unique_scores[j] <= chacha[i]:
            ans.append(j+1)
            i -= 1
        else:
            j += 1

    return reversed(ans)
scores_count = int(input())
scores = list(map(int, input().rstrip().split()))
chacha_count = int(input())
chacha = list(map(int, input().rstrip().split()))
result = climbing(scores, chacha)
print(result)
```

C

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int n,i,j,t,scores_i,k;
    scanf("%i", &n);
    int *scores = malloc(sizeof(int) * n);
    scanf("%i",&scores[0]);
    for (scores_i = 1,k=1; k < n;k++ )
    {
       scanf("%i",&t);
       if(t !=scores[scores_i-1])
       {
           scores[scores_i]=t;
           scores_i++;
       }
     }
    n=scores_i;
    int m,rank;
    j=n-1;

    scanf("%i", &m);
    int *alice = malloc(sizeof(int) * m);
    for (int alice_i = 0; alice_i < m; alice_i++) {
       scanf("%i",&alice[alice_i]);
    }
    for(i=0;i<m;i++)
    {
        while(j>=0 && alice[i]>scores[j])
            j--;
        if(j==-1)
            rank=1;
        else if(alice[i] == scores[j])
            rank=j+1;
        else if(alice[i] < scores[j])
            rank=j+2;
        printf("%d\n",rank);
    }
    return 0;
}
```

QUESTION 5

Problem Statement:- You will be given an array of integers and a target value. Determine the number of pairs of array elements that have a difference equal to a target value.

For example, given an array of [1, 2, 3, 4] and a target value of 1, we have three values meeting the condition:

2-1 = 1

3-2 = 1

4-3 = 1.

Function Description

Write a function *pairs*. It must return an integer representing the number of element pairs having the required difference.

pairs has the following parameter(s):

- *k*: an integer, the target difference
- *arr*: an array of integers

Input Format

- The first line contains two space-separated integers n and k, the size of arr and the target value.
- The second line contains n space-separated integers of the array arr.

Sample Input

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

Sample Output

```
2
```

PROGRAMS:

Python

```
def pairs(k, arr):
    q = 0
    n = len(arr)
    arr = set(arr)
    h = len(arr) – n
    for i in arr:
        if i+k in arr:
            q+=1
    return q+h
nk = input().split()
n = int(nk[0])
k = int(nk[1])
arr = list(map(int, input().rstrip().split()))
result = pairs(k, arr)
print(result)
```

C

```
#include <stdio.h>
int countPairsWithDiffK(int arr[], int n, int k)
{
    int count = 0;
    for (int i = 0; i < n; i++)
    {
        for (int j = i+1; j < n; j++)
        {
            if (arr[i] - arr[j] == k || arr[j] - arr[i] == k )
            {
                count++;
            }
        }
    }
    return count;
}
int                                                                                                    main()
{
    int arr[] =  {1, 5, 3, 4, 2};
    int n = sizeof(arr)/sizeof(arr[0]);
    int k = 3, result;
    result = countPairsWithDiffK(arr, n, k);
    printf("%d",result);
    return 0;
}
```

QUESTION 6

Problem Statement:- A jail has a number of prisoners and a number of treats to pass out to them. Their jailer decides the fairest way to divide the treats is to seat the prisoners around a circular table in sequentially numbered chairs. A chair number will be drawn from a hat. Beginning with the prisoner in that chair, one candy will be handed to each prisoner sequentially around the table until all have been distributed.

The jailer is playing a little joke, though. The last piece of candy looks like all the others, but it tastes *awful*. Determine the chair number occupied by the prisoner who will receive that candy.

For example, there are 4 prisoners and 6 pieces of candy. The prisoners arrange themselves in seats numbered 1 to 4. Let's suppose two are drawn from the hat. Prisoners receive candy at positions 2,3,4,1,2,3. The prisoner to be warned sits in chair number 3

Function Description

Write a function *saveThePrisoner*. It should return an integer representing the chair number of the prisoner to warn.

saveThePrisoner has the following parameter(s):

- *n*: an integer, the number of prisoners
- *m*: an integer, the number of sweets
- *s*: an integer, the chair number to begin passing out sweets from

Input Format

- The first line contains an integer t, denoting the number of test cases.
- The next t lines each contain 3 space-separated integers:
  - – : n the number of prisoners
  - – : m the number of sweets
  - – : s the chair number to start passing out treats at

Output Format: For each test case, print the chair number of the prisoner who receives the *awful treat* on a new line.

Sample Input

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

Sample Output

```
2
3
```

PROGRAMS:

Python

```
def saveThePrisoner(n, m, s):
    if (m + s - 1) % n == 0:
        return n
    else:
        return (m+s-1) % n
t = int(input())
for t_itr in range(t):
    nms = input().split()
    n = int(nms[0])
    m = int(nms[1])
    s = int(nms[2])

    result = saveThePrisoner(n, m, s)
    print(result)
```

C

```
#include <stdio.h>
int main()
{
    int t,j,i,count=0;
    long int ncr;
    long int result,diff;
    scanf("%d",&t);
    long int n[t];
    long int m[t];
    long int s[t];
    for(i=0;i<t;i++)
    {
        scanf("%ld %ld %ld",&n[i],&m[i],&s[i]);
    }
    for(i=0;i<t;i++)
    {
        count=(n[i]-s[i])+1;
        if(count>=m[i])
        {
            result=(s[i]+m[i])-1;
            printf("%ld\n",result);
        }
        if(count<m[i])
        {
            diff=m[i]-count;
            while(diff>n[i])
            {
                diff=diff-n[i];
            }
            printf("%ld\n",diff);
        }
    }
    return 0;
}
```

QUESTION 7
Write a program to find the count of numbers which consists of unique digits.

Input:
Input consist of two Integer lower and upper value of an range

Output:
Output consists of single line, print the count of unique digits in given range. Else Print "**No Unique Number**"

Solution:

Input -
10
15

PROGRAMS:
C++

```
#include<bits/stdc++.h>

using namespace std;

void printUnique(int l, int r)

{
   int count=0;
   for (int i=l ; i<=r ; i++)

   {
      int num = i;
      bool visited[10] = {false};

      while (num)
      {

         if (visited[num % 10])
            break;

         visited[num%10] = true;

         num = num/10;
      }

      if (num == 0)
         count++;
   }
```

```
    if(count>0)
     cout<<count;
    else
        cout<<"No Unique Number";
}

int main()
{
    int l,r;
    cin>>l>>r;
    printUnique(l, r);
    return 0;
}
```

Output

```
5
```

QUESTION 8

There is a range given n and m in which we have to find the count all the prime pairs whose difference is 6. We have to find how many sets are there within a given range.

Output:

Output consists of single line, print the count prime pairs in given range. Else print"**No Prime Pairs**".

Constraints:

2<=n<=1000
n<=m<=2000

Sample Input:

4
30

Output:

6

Explanation:

(5, 11) (7, 13) (11, 17) (13, 19) (17, 23) (23, 29) . we have 6 prime pairs.

Solution:

Input -

101
500

PROGRAMS:

C++

```
#include <bits/stdc++.h>
using namespace std;

void count_prime(int l, int r)
{

    int count=0;
    bool prime[r + 1];
    memset(prime, true, sizeof(prime));

    for (int p = 2; p * p <= r; p++) {
        if (prime[p] == true) {
            for (int i = p * 2; i <= r; i += p)
                prime[i] = false;
        }
    }
    for (int i = l; i <= r - 6; i++)
```

```
        if (prime[i] && prime[i + 6])
            count++;

    if(count>0)
        cout<<count;
    else
        cout<<"No Prime Pairs";
}

int main()
{

    int n,m;
    cin>>n>>m;
    count_prime(n, m);
    return 0;
}
```

Output

```
30
```

QUESTION 9
Write a program to print all the combinations of the given word with or without meaning (when unique characters are given).

Sample Input: abc

Output:
```
abc
acb
bac
bca
cba
cab
```

Solution:
Input: hai

PROGRAMS:
C++
```
#include<bits/stdc++.h>
using namespace std;
void permute(string a, int l, int r)
{
    if (l == r)
        cout<<a<<endl;
    else
    {
        for (int i = l; i <= r; i++)
        {
            swap(a[l], a[i]);
            permute(a, l+1, r);
            swap(a[l], a[i]);
        }
    }
}
int main()
{
    string str;
    cin>>str;
    int n = str.size();
    permute(str, 0, n-1);
    return 0;
}
```

Output
```
hai hia ahi aih iah iha
```

QUESTION 10

Bastin once had trouble finding the numbers in a string. The numbers are distributed in a string across various test cases. There are various numbers in each test case you need to find the number in each test case. Each test case has various numbers in sequence. You need to find only those numbers which do not contain 9. For eg, if the string contains "hello this is alpha 5051 and 9475".You will extract 5051 and not 9475. You need only those numbers which are consecutive and you need to help him find the numbers. Print the largest number.
Note: Use long long for storing the numbers from the string.

Input:
The first line consists of **T** test cases and next **T** lines contain a string.

Output:
For each string output the number stored in that string if various numbers are there print the largest one. If a string has no numbers print -1.

Constraints:
1<=**T**<=100
1<=**|S|**<=10000

Example:
Input:
1
This is alpha 5057 and 97
Output:
5057

Solution:

Input -
1
dream job 100 and 101

PROGRAMS:
C++

```
#include<bits/stdc++.h>
using namespace std;

int main()
{
  int t;
  cin >> t;
  cin.ignore();

  while(t--)
  {
    string s;
    getline(cin, s);
    int n = s.length();
    int n9 = 0;
    string res="", num = "";
    for(int i = 0; i < n; i++)
    {
      n9 = 0;
      num = "";
      while(s[i] >= '0' && s[i] <= '9')
      {
        if(s[i] == '9')
          n9 = 1;

        num = num + s[i];
        i++;
      }

      if(!n9 && num != "")
      {
        long long a = stoll(num);
        long long b = -1;

        if(res != "")
          b = stoll(res);

        if(a> b)
          res = num;
      }
    }
    if(res == "")
      cout << "-1";
    else
      cout <<res << endl;
  }
  return 0;
}
```

Output

```
101
```