using System;

using System.Collections.Generic;

using System.Linq;

namespace Assignment1\_Spring2021

{

class Program

{

static void Main(string[] args)

{

//Question 1

Console.WriteLine("Q1 : Enter the number of rows for the traingle:");

int n = Convert.ToInt32(Console.ReadLine());

printTriangle(n);

Console.WriteLine();

//Question 2:

Console.WriteLine("Q2 : Enter the number of terms in the Pell Series:");

int n2 = Convert.ToInt32(Console.ReadLine());

printPellSeries(n2);

Console.WriteLine();

//Question 3:

Console.WriteLine("Q3 : Enter the number to check if squareSums exist:");

int n3 = Convert.ToInt32(Console.ReadLine());

bool flag = squareSums(n3);

if (flag)

{

Console.WriteLine("Yes, the number can be expressed as a sum of squares of 2 integers");

}

else

{

Console.WriteLine("No, the number cannot be expressed as a sum of squares of 2 integers");

}

//Question 4:

int[] arr = { 3, 1, 4, 1, 5 };

Console.WriteLine("Q4: Enter the absolute difference to check");

int k = Convert.ToInt32(Console.ReadLine());

int n4 = diffPairs(arr, k);

Console.WriteLine("There exists {0} pairs with the given difference",n4);

//Question 5:

List<string> emails = new List<string>();

emails.Add("dis.email + bull@usf.com");

emails.Add("dis.e.mail+bob.cathy@usf.com");

emails.Add("disemail+david@us.f.com");

int ans5 = UniqueEmails(emails);

Console.WriteLine("Q5");

Console.WriteLine("The number of unique emails is " + ans5);

//Quesiton 6:

string[,] paths = new string[,] { { "London", "New York" }, { "New York", "Tampa" },

{ "Delhi", "London" } };

string destination = DestCity(paths);

Console.WriteLine("Q6");

Console.WriteLine("Destination city is " + destination);

}

/// <summary>

///Print a pattern with n rows given n as input

///n – number of rows for the pattern, integer (int)

///This method prints a triangle pattern.

///For example n = 5 will display the output as:

/// \*

/// \*\*\*

/// \*\*\*\*\*

/// \*\*\*\*\*\*\*

/// \*\*\*\*\*\*\*\*\*

///returns : N/A

///return type : void

/// </summary>

/// <param name="n"></param>

private static void printTriangle(int n)

{

try

{

int i; //the row number  
 for(i=1;i<=n;i++){

//the number of spaces for each row is n-i  
 for(int space=0;space<n-i;space++)  
 Console.Write(" ");  
 //the number of \* for each row is 2\*i-1  
 for(int j=0;j<2\*i-1;j++)  
 Console.Write("\*");  
   
 Console.Write("\n");  
   
 }

}

catch (Exception)

{

throw;

}

}

//self-reflection: number of spaces and stars for each row is related to the row number

//time taken:0.06s

/// <summary>

///<para>

///In mathematics, the Pell numbers are an infinite sequence of integers.

///The sequence of Pell numbers starts with 0 and 1, and then each Pell number is the sum of twice of the previous Pell number and

///the Pell number before that.:thus, 70 is the companion to 29, and 70 = 2 × 29 + 12 = 58 + 12. The first few terms of the sequence are :

///0, 1, 2, 5, 12, 29, 70, 169, 408, 985, 2378, 5741, 13860,…

///Write a method that prints first n numbers of the Pell series

/// Returns : N/A

/// Return type: void

///</para>

/// </summary>

/// <param name="n2"></param>

private static void printPellSeries(int n2)

{

try{

//for the first two items  
 if(n2<=2){  
 if(n2==1)Console.Write("0");  
 if(n2==2)Console.Write("0 1");  
 }   
 else{

//still need to print the first two  
 Console.Write("0 1 ");  
 int P0 = 0,P1 = 1,Pn;

//calculate third to n items and print  
 for (int i = 3; i <= n2; i++) {   
 Pn = 2 \* P1 + P0;   
 Console.Write(Pn);  
 Console.Write(" ");  
 P0 = P1;   
 P1 = Pn;   
 }   
 }  
 }

catch (Exception)

{

throw;

}

}

//self-reflection: still need to take care of the first two items when n2>=3

//time taken:0.06s

/// <summary>

///Given a non-negative integer c, decide whether there're two integers a and b such that a^2 + b^2 = c.

///For example:

///Input: C = 5 will return the output: true (1\*1 + 2\*2 = 5)

///Input: C = 3 will return the output : false

///Input: C = 4 will return the output: true

///Input: C = 1 will return the output : true

///Note: You cannot use inbuilt Math Class functions.

/// </summary>

/// <param name="n3"></param>

/// <returns>True or False</returns>

private static bool squareSums(int n3)

{

try

{

//the equation (n3/2+1)^2>n3 is true for all n3 in math

//there is no need to go through the integers>n3/2

//however, in the program, there is one exception

//n3=1, n3/2 equals 0 in the program, so first we need to see if n3/2 equals 0

long i=0,mid=n3/2,temp;

        if (mid==0) return true;

        while(i<=mid){

            temp = i\*i+mid\*mid;

            if(temp>n3) mid--;

            else if(temp<n3) i++;

            else return true;

        }

        return false;

}

catch (Exception)

{

throw;

}

}

//self-reflection: the program turns out to be shorter than I thought.

//time taken is shown in the screenshot

/// <summary>

/// Given an array of integers and an integer n, you need to find the number of unique

/// n-diff pairs in the array.Here a n-diff pair is defined as an integer pair (i, j),

///where i and j are both numbers in the array and their absolute difference is n.

///Example 1:

///Input: [3, 1, 4, 1, 5], k = 2

///Output: 2

///Explanation: There are two 2-diff pairs in the array, (1, 3) and(3, 5).

///Although we have two 1s in the input, we should only return the number of unique

///pairs.

///Example 2:

///Input:[1, 2, 3, 4, 5], k = 1

///Output: 4

///Explanation: There are four 1-diff pairs in the array, (1, 2), (2, 3), (3, 4) and

///(4, 5).

///Example 3:

///Input: [1, 3, 1, 5, 4], k = 0

///Output: 1

///Explanation: There is one 0-diff pair in the array, (1, 1).

///Note : The pairs(i, j) and(j, i) count as same.

/// </summary>

/// <param name="nums"></param>

/// <param name="k"></param>

/// <returns>Number of pairs in the array with the given number as difference</returns>

private static int diffPairs(int[] nums, int k)

{

try

{

// in this case , there is no pair

if (nums.Length == 1) return 0;

// sort the array & get the length

Array.Sort(nums);

int L = nums.Length;

int start = 0, end = 1;

var map = new HashSet<int>();

int diff = 0;

while (start < L && end <L)

{

diff = nums[end] - nums[start];

if (diff > k)

{

start++;

if (start == end)

end++;

}

else if (diff < k)

end++;

else

{

// check if there is duplicate & and store

if (!map.Contains(nums[start]))

map.Add(nums[start]);

// since the array has been sorted and we’ve found the pair for the current nums[start], we can move on

end++;

start++;

}

}

return map.Count;

}

catch (Exception e)

{

Console.WriteLine("An error occured: " + e.Message);

throw;

}

}

//time taken is shown in the screenshot

/// <summary>

/// An Email has two parts, local name and domain name.

/// Eg: rocky @usf.edu – local name : rocky, domain name : usf.edu

/// Besides lowercase letters, these emails may contain '.'s or '+'s.

/// If you add periods ('.') between some characters in the local name part of an email address, mail sent there will be forwarded to the same address without dots in the local name.

/// For example, "bulls.z@usf.com" and "bullsz@leetcode.com" forward to the same email address. (Note that this rule does not apply for domain names.)

/// If you add a plus('+') in the local name, everything after the first plus sign will be ignored.This allows certain emails to be filtered, for example ro.cky+bulls @usf.com will be forwarded to rocky@email.com. (Again, this rule does not apply for domain names.)

/// It is possible to use both of these rules at the same time.

/// Given a list of emails, we send one email to each address in the list.Return, how many different addresses actually receive mails?

/// Eg:

/// Input: ["dis.email+bull@usf.com","dis.e.mail+bob.cathy@usf.com","disemail+david@us.f.com"]

/// Output: 2

/// Explanation: "disemail@usf.com" and "disemail@us.f.com" actually receive mails

/// </summary>

/// <param name="emails"></param>

/// <returns>The number of unique emails in the given list</returns>

private static int UniqueEmails(List<string> emails)

{

try

{

// split every address by “@”

// split those that contain “+”

// just replace “.” with blank

// use distinct() to get the number of different email addresses

return emails.Select(element=>element.Split('@')).Select(element=>$"{element[0].Split('+')[0].Replace(".","")}@{element[1]}").Distinct().Count();

}

catch (Exception e)

{

Console.WriteLine(e.Message);

throw;

}

}

//time taken is shown in the screenshot

/// <summary>

/// You are given the array paths, where paths[i] = [cityAi, cityBi] means there exists a direct path going from cityAi to cityBi. Return the destination city, that is, the city without any path outgoing to another city.

/// It is guaranteed that the graph of paths forms a line without any loop, therefore, there will be exactly one destination city.

/// Example 1:

/// Input: paths = [["London", "New York"], ["New York","Tampa"], ["Delhi","London"]]

/// Output: "Tampa"

/// Explanation: Starting at "Delhi" city you will reach "Tampa" city which is the destination city.Your trip consist of: "Delhi" -> "London" -> "New York" -> "Tampa".

/// Input: paths = [["B","C"],["D","B"],["C","A"]]

/// Output: "A"

/// Explanation: All possible trips are:

/// "D" -> "B" -> "C" -> "A".

/// "B" -> "C" -> "A".

/// "C" -> "A".

/// "A".

/// Clearly the destination city is "A".

/// </summary>

/// <param name="paths"></param>

/// <returns>The destination city string</returns>

private static string DestCity(string[,] paths)

{

try{

for(int j=0;j<paths.GetLength(0);j++)

{

int d=0;

for(int i=0;i<paths.GetLength(0);i++)

{

if(paths[i,0]==paths[j,1])// check if there is a match for paths[j,1]

{

d=1;

break;// we found the match, then continue to find the next city in the path

}

}

// finally, there is no match for paths[j,1], then it’s our destination

if(d==0)

{

return paths[j,1];

}

}

return "";//error occurs without this line

}

catch (Exception)

{

throw;

}

}

}

}

//time taken is shown in the screenshot