|  |  |  |
| --- | --- | --- |
|  |  | **ISM 6225**  **Distributed Information systems** |
| Prof Agrawal TA: Phani Bhushan Kolla | | |

Assignment 1 – Programming Introduction

Primary objective: Develop familiarity with essential programming constructs

Secondary objective: develop comfort with using the IDE and GitHub

*Estimated time: 25 hours*

## Introduction

Full-stack application development is an essential skill needed to succeed and even survive in business analytics and/ or information systems roles, especially as AI takes over many rudimentary tasks formerly performed by analysts. This assignment introduces the essential programming constructs such as variables, selection, loops, methods, and arrays used to build such applications. Specifically, this assignment avoids the use of API methods and object-oriented programming. Those tasks are left for later assignments. This assignment also does not check for efficiency in program implementation. That is something you will develop over a lifetime in the profession. Rather, the focus is on simple programming exercises for students to learn basic industry best practices. One design goal for this assignment was to focus tightly on introductory programming structures, with a low probability that students would find ready-to-use solutions online.

This is an individual assignment, to allow every student to develop the necessary skills to become a productive contributor to project teams in this class and beyond.

## Activity

Use a programming language of your choice to define methods to do the operations specified in the method signatures and hints below. The methods are listed in the recommended sequence of development. A starter Program.cs file is included in the appendix.

## Submission

Push the code to GitHub and submit the URL to Canvas. Also, get the output from a sample run that shows the use of all required methods and upload/push a screenshot to GitHub. This serves as a quick check. Submit your self-reflection as a comment on the assignment.

## Grading scheme

Each method carries 1 point. You will be graded on the following aspects for each question:

Logic (including appropriate organization of logic into methods) : 0.5

Handling all reasonable corner cases : 0.2

Descriptive comments explaining the logic to reviewer : 0.2

Self-reflection (time taken, learning, and recommendations) : 0.1

## Method specifications

**QUESTION 1:**

/\*

Print a pattern with n rows given n as input

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

\* summary : This method prints a triangle pattern.

\* For example n = 5 will display the output as:

\*

**\***

**\*\*\***

**\*\*\*\*\***

**\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\***

\*

\* returns : N/A

\* return type : void

\*/

private static void PrintTriangle(int n)

**QUESTION 2:**

/\*

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

\*/

private static void PrintPellSeries(int n2)

**QUESTION 3:**

/\* Given a non-negative integer c, decide whether there're two integers a and b such \* that a2 + b2 = c.

\* For example:

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

\* Input: A = 3 will return the output : false

\* Input: A = 4 will return the output: true

\* Input: A = 1 will return the output : true

\*

\* Note: You cannot use inbuilt Math Class functions.

\* returns : Boolean Value

\* return type : bool

\*/

private static bool SquareSums(int A)

**QUESTION 4:**

/\* 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.

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

**QUESTION 5:**

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

Eg: [rocky@usf.edu](mailto: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

Returns integer

Return type : int \*/

private static int UniqueEmails(string[] emails)

**Question 6:**

/\* 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".

Return type : string \*/

private static string DestCity(string[,] paths)

Appendix:

Program.cs

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

{

// write your code here

}

catch (Exception)

{

throw;

}

}

/// <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

{

// write your code here.

}

catch (Exception)

{

throw;

}

}

/// <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: A = 3 will return the output : false

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

///Input: A = 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

{

// write your code here\

return false;

}

catch (Exception)

{

throw;

}

}

/// <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

{

// write your code here.

return 0;

}

catch (Exception e)

{

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

throw;

}

}

/// <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

{

// write your code here.

return 0;

}

catch (Exception e)

{

Console.WriteLine(e.Message);

throw;

}

}

/// <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

{

// write your code here.

return "";

}

catch (Exception)

{

throw;

}

}

}

}