|  |  |  |  |
| --- | --- | --- | --- |
| **МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  **КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка**  **ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**  **Кафедра програмних систем і технологій**  **Дисципліна**  **«Кросплатформне програмування»**  **Лабораторна робота 2** | | | |
| **Виконав:** | Кивлюк Юрій Олегович | **Перевірив**: | Петрівський В.Я |
| Група | ІПЗ-41 | Дата перевірки |  |
| Форма навчання | денна | Оцінка |  |
| Спеціальність | 121 |
| 2023 | | | |

**Варіант – 62**

Посилання на виконану роботу – [Github](https://github.com/yura-kiv/crossplatform-uni/tree/master/lab_2).

**Умова завдання:**

Розглянемо числову послідовність, що спочатку складається з двох одиниць: 1, 1. Далі на кожному наступному кроці вставлятимемо між сусідніми елементами їх суму. У прикладі елементи, що додаються, виділені:

|  |  |
| --- | --- |
| Номер кроку | Послідовність |
| 0 | 1, 1 |
| 1 | 1, 2, 1 |
| 2 | 1, 3, 2, 3, 1 |
| 3 | 1, 4, 3, 5, 2, 5, 3, 4, 1 |

Потрібно написати програму, яка підрахує суму членів послідовності, побудованої за K кроків.

Вхідні дані:

Вхідний файл INPUT.TXT містить одне натуральне число K (0 ≤ K ≤ 100) – номер останнього кроку.

Вихідні дані:

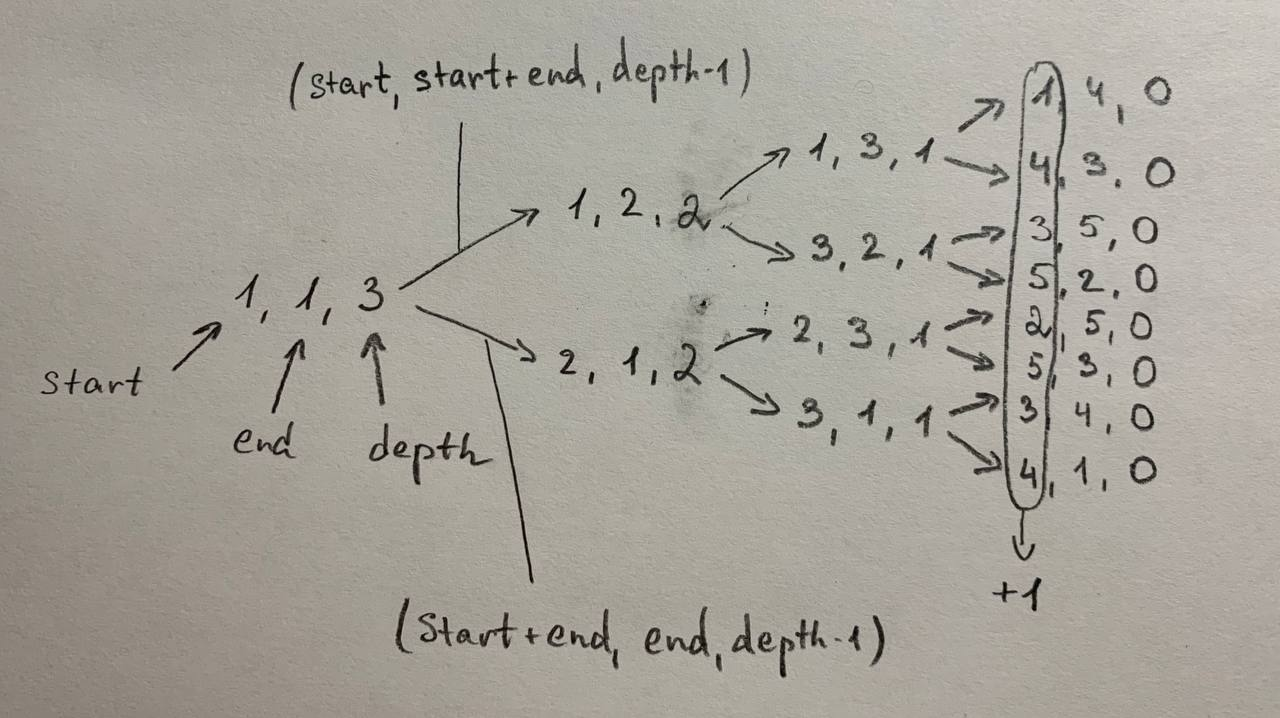
Вихідний файл OUTPUT.TXT повинен містити одне натуральне число – суму елементів послідовності, побудованої за кроків K.

Приклади:

|  |  |  |
| --- | --- | --- |
| № | INPUT.TXT | OUTPUT.TXT |
| 1 | 3 | 28 |
| 2 | 10 | 59050 |

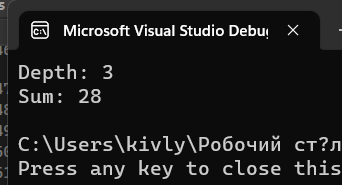
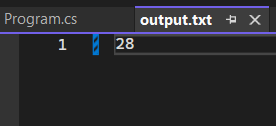
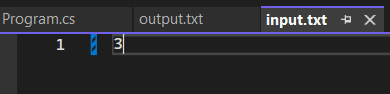
**Виконання завдання:**

Принцип роботи коду:

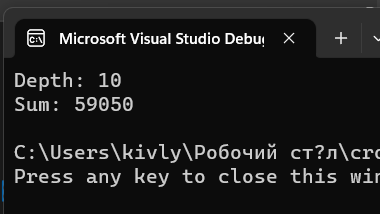
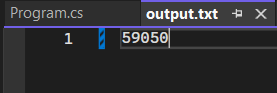
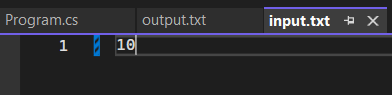


Результат виконання:

1)

2)

Реалізований код:

class Program

{

const string INPUT\_FILE = "../../../input.txt";

const string OUTPUT\_FILE = "../../../output.txt";

static int getDepthFromFile()

{

int depth = -1;

try

{

string inputText = File.ReadAllText(INPUT\_FILE);

if (int.TryParse(inputText, out depth) && depth >= 0 && depth <= 100)

{

return depth;

}

else

{

Console.WriteLine("Invalid value in file. Only integer values between 0 and 100 are valid.");

return -1;

}

}

catch

{

Console.WriteLine("There was a problem reading the file..");

return -1;

}

}

static BigInteger GetSequenceSum(int start, int end, int depth, BigInteger sum)

{

if (depth == 0)

{

return sum + start;

}

sum = GetSequenceSum(start, start + end, depth - 1, sum);

sum = GetSequenceSum(start + end, end, depth - 1, sum);

return sum;

}

static void Main()

{

int depth = getDepthFromFile();

if (depth == -1) return;

Console.WriteLine("Depth: " + depth);

BigInteger sum = GetSequenceSum(1, 1, depth, 0) + 1;

Console.WriteLine("Sum: " + sum);

File.WriteAllText(OUTPUT\_FILE, sum.ToString());

}

}