Issue Date: February 10, 2025
Start Time: 2:00 PM
Submission Time: 3:30 PM

Total Marks: 50

Data Structures and Algorithms Lab
CS-F23

LAB-02

The objective of this lab is to:
Determine time complexity, growth functions & big oh notation of code snippets. Practice writing code in O(n) time complexity.

Instructions:
1) Follow the question instructions very carefully, no changes in function prototypes are allowed.
2) You are **not allowed** to use any external libraries.
3) Anyone caught in an act of plagiarism would be awarded an "F" grade in this Lab.

Task 01(Time Complexity Analysis) [10 Marks]

Consider the following code snippets. You are required to write the growth function and specify the Big-Oh of each of the following codes:

1-

```
for(int i = 0; i < n/2; i++)
{
   for(int j = n; j >= 0; j--)
   {
         for(int k =0; k<n; k++)
         {

         }
   }
}
```

2-

```
for (int i = 0; i < n / 2; i++)
{
    for (int j = 1; j < n; j *= 2)
    {
        count++;
    }
}
```

Task 02 (Sum of Array Except Self) [10 Marks]

You are given an integer array nums of size n. Your task is to return another array result such that result[i] is equal to the sum of all elements of nums except nums[i].

Example:

Input: nums = {1, 2, 3, 4}
Result: result = {9, 8, 7, 6}

Note: In order to get full marks, write code that has time complexity O(n).

Function Prototype :

int * sumExceptSelf(int * nums, int size)

Task 03 (Maximum Enemy Forts Captured) [15 Marks]

You are given a 0-indexed integer array forts of length n representing the positions of several forts. forts[i] can be -1, 0, or 1 where:

- **-1 represents there is no fort at the i^{th} position.**
- **0 indicates there is an enemy fort at the i^{th} position.**
- **1 indicates the fort at the i^{th} the position is under your command.**

Now you have decided to move your army from one of your forts at position i to an empty position j such that:

- **0 <= i, j <= n - 1**
- **The army travels over enemy forts only. Formally, for all k where min(i,j) < k < max(i,j), forts[k] == 0.**

While moving the army, all the enemy forts that come in the way are captured.
Return *the maximum number of enemy forts that can be captured*. In case it is impossible to move your army, or you do not have any fort under your command, return 0.

Example:

Input: forts = {1,0,0,-1,0,0,0,0,1}
Output: 4

Explanation:
- Moving the army from position 0 to position 3 captures 2 enemy forts, at 1 and 2.
- Moving the army from position 8 to position 3 captures 4 enemy forts.
Since 4 is the maximum number of enemy forts that can be captured, we return 4.

Note:

Find solution in linear time complexity i.e O(n).

Task 04 (Find an element in matrix) [15 Marks]

Given an $N \times N$ matrix of integers where each row is sorted in increasing order from left to right and each column is sorted in increasing order from top to bottom, design an algorithm that determines if a number X is present in the matrix. The algorithm should have a worst-case time complexity of O(N).

Task 05 (Sparse Matrix) [10 Marks]

Element.h

```
class Element
{
	int row;
	int column;
	int value;
public:
	Element(int, int, int);
	Element();
	void setRow(int);
	void setColumn(int);
	void setValue(int);
	int getRow();
	int getColumn();
	int getValue();
};
```

Sparse.h

```
#include"Element.h"
class SparseMatrix
{
	Element* data;
	int NOE;
	int noOfRows;
	int noOfCols;
	bool isUniqueCell(int, int);
	void reSize();
public:
	SparseMatrix();
	SparseMatrix(int, int);
	~SparseMatrix();
	int getIndexValue(int, int) const;
	void insert(int, int, int);
	SparseMatrix add(const SparseMatrix&) const;
	SparseMatrix multiply(const SparseMatrix& other) const;
	void print();
};
```