HW-1: CSC241_Sec2 (Data Structure and Algorithm)

Due Date: 02/17/2023 (11.59:59 PM)

Special Instruction: Your HW will be graded based on correctness and clarity. Keep you answer precise and to the point. If any question ask for justification of your answer/claim, you may receive a 0 if you merely provide an answer without justification. ***Your answer must need to be printed. Handwritten submissions will not be evaluated***. All sub questions carry equal weights unless specified otherwise. Finally, please check the HW rules at the end.

What to do: Reach to me ASAP if you have any confusion and/or have any emergency that may deter you to submit HW on time. Never hesitate to ask me if any of the previously discussed topics is unclear and you need some more discussion.

What not to do:

1. Ask to verify your solution
2. Ask to debug/analyze your code

Question 1 (22 Points): Consider the code below, which calculates the sum of all numbers from 1 to n. Here, n is a positive integer. Write the recursive version of this code. ***Test your code for these n values: 25, 50, 67, 100.***

```
int getSum(int n)
{
	int sum = 0;
	for (int i = 1; i<= n; i++)
	{
		sum = sum + i;
	}

	return sum;
}
```

Question 2 (25 Points): In this problem, you need to create a singly linked list containing N integers ($N \geq 10$) with at least 3 or more duplicate items. The head and tail must have to be a duplicate item, and insert more duplicate items at other random position (see the sample inputs below for clarity, red colored items are duplicate). Now,

- Delete the last occurrence of any duplicate items and print the updated list.
- Print the size of the updated linked list

Sample Input 1: 54 63 21 54 17 23 37 118 21 107 37
Sample Output 1: 54 63 21 17 23 37 118 107
Sample Output 1: There are 8 items on the list.

Sample Input 2: 54 63 21 17 23 21 118 18 10 54
Sample Output 2: 54 63 21 17 23 118 18 10

Sample Output 1: There are 8 items on the list.

Special instructions for Question 1 and 2: Please upload all of the following to D2L separately (not as one zip file):

- A .pdf or .docx that contains your written answers. This file must show your code (also requested below). Finally, attach a screenshot from your output console/terminal.
- The Java files (or other files that contain your code).

Question 3 (15 Points): Consider the following definition for a function SplitNum(int N).

$$SplitNum(1) = 1$$

$$SplitNum(n) = SplitNum((n+1)/2) + SplitNum(n/2)$$

(i) According to the definition, what is the value of **SplitNum(9)**? Show intermediate steps while calculating the final value.

(ii) We wrote the following wrong recursive pseudocode to implement $SplitNum(n)$. Modify this code snippet so that it works properly.

```
int splitNum(int num){
      if(num >= 1) return num;
      else{
            return splitNum(num/2+(num+1)/2);
      }
}
```

Question 4 (16 Points): Consider the following code snippets and answer the following questions.

- What will be the output if we call "*DigitSum(81406)*"?
- Write the recursive version of "*DigitSum(num)*". Unlike Question 1 and 2, it does not have to be fully functional java code. A logic with proper/adequate comments will be sufficient.

```
public static int DigitSum(long num){
     int sum = 0;
     while (num > 0)
     {
         sum += num % 10;
         num = num / 10;
     }
     return sum;
}
```

Question 5 (12 Points): Assume a classroom is divided into multiple columns, each containing several students. You pick a random student and ask him/her two simple questions:

- How many students are directly behind you in your column?
- How many students are directly in front of you in your column?

All students can see only the person sitting in front (and behind) of them. However, they can communicate with the person in front or behind them. Considering this scenario, for any student, can you write a recursive method that counts the number of people sitting in front or behind that student?

Question 6 (10 Points): Consider the following questions and determine whether they are True or False.

I. In a linked list, the first node never refers to NULL.
II. A linked list uses dynamic type of memory allocation.
III. Accessing the 5^{th} element Vs accessing the 500^{th} element takes same amount of time in an array.
IV. Accessing the 5^{th} element Vs accessing the 500^{th} element takes same amount of time in a linked list.
V. Static Memory Allocation is better than dynamic memory allocation when we want to utilize memory efficiently.

Rules for ALL HW:

1. If any programming problem is given, the code must be written by yourself. DO NOT copy code from anywhere else.

2. You can discuss the problem sets and study together in group, but when it comes to formulating/writing solutions you must work alone independently; i.e., you should be able to explain your answer clearly to anyone else. Note that this says discuss in group — copying homework solutions from another student, from the Internet, solution sets of friends who have taken this course or one similar to it previously, or other sources will be considered **cheating** and referred to the university. *At the beginning of each submission, you should explicitly list the people you worked with.*