Name and

AMERICAN UNIVERSITY OF ARMENIA
College of Science and Engineering
CS 121 Data Structures and Algorithms

MIDTERM 1 EXAM

Date:	Tuesday, October 18 2016
Starting time:	09:00
Duration:	1 hour 15 min
Attention:	**ANY TYPE OF COMMUNICATION IS STRICTLY PROHIBITED**

Please write down your name and ID# at the top of all used pages

16/24

Problem 1: Consider below two recursive expressions:

$$a_n = 1 + a_1*b_1 + a_2*b_2 + a_3*b_3 + \ldots + a_{n-1}*b_{n-1}$$
$$b_n = 1 + 2*b_1 + 2*b_2 + 2*b_3 + \ldots + 2*b_{n-1} - b_{n-1}*b_{n-1}$$

The base cases are: $a_1 = b_1 = 1$.

Write an optimal C++ function or Java method that takes as its argument an int index ***int n*** and returns a_n.

```
public static int rec(int n){
    if(n==1){
        a=1; b=1;
        return a;
    }
    else
    b = 2 * b + b;
    a = a * b + rec(n-1);
    return ?
}
```

2

Use the backside, if needed

DS.MT1.181016.M153

Name and

Problem 3: Consider a text that can contain four types of braces: ( ), [ ], { } and < >. The braces are balanced, if the following two conditions hold:

1. Each time a closing brace is encountered, it matches an already encountered corresponding opening brace.
2. At the end of the text, each opening brace is matching the respective closing one.

For example, the braces are balanced in a text *{ab(c[d])e}*, but not balanced in *{ab(c))}*.

Write a C++ function ***bool balanced_brackets(string text)*** or a Java method ***public static boolean balancedBrackets(String text)*** that take as the argument a string text and check, if the brackets of all four types are balanced or not. Use ***stack<char>*** in C++ or ***Stack<Character>*** in Java.

```
public stat - boolean balanced_brackets(String text){
    Stack <char> stack = new stack;
    boolean balanced = true;
    index = 0;
    while (balanced && index < text.length){
    text.charAt(i) = character;
    i ++;
      if (character == "(" || "[" || "{" || "<")
         stack.push(character);
      else if (character == ")" || "]" || "}" || ">")
         if(! stack.isEmpty)
            stack.pop();   pop the counterpart

      else
        balanced == false; }
      if (!stack.isempty() && balanced)
         return true;
      else
        return false;
      }
```

5

Use the backside, if needed