/

Name and ID#: [redacted]

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**

9/21

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

Problem 1: Consider below two recursive expressions:

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

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.

```
~~int function(int a, int b) {~~
~~  if(b == 1) {~~
~~     return 1~~
~~  }~~
~~  else {~~
~~    return~~

int funktion(int a, int b~~[illegible]~~, int n) {
  if(n == 1) {
     ~~a=1~~ return 1;
  }
  else {
    return a·b + funltion(a - a_{n-1}·b_{n-1}; b - 2b_{n-1}; n--);
  }
```

\ /

wrong....!

4

Use the backside, if needed

DS.MT1.181016.M156

Name and ID#: [redacted]

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.

```
String new = "";
public static boolean balancedBrackets(string text) {
  for(int i=0; i<text.length(); i++) {
    if (text.charAt(i)=='{' || text.charAt(i)=='}' || text.charAt(i)='[' ... || text.charAt(i)=')') {
      new~~.concat(text.charAt~~ += text.charAt(i);
    }
  }
//now we have a literally "new" string consisting only from brackets
while(new) {
  for(int i=0; i<new.length-2; i++) {
    if(new.charAt(i)== new.charAt(i+1)) {
      ~~String str = new.substring(0,~~ String str = "";
      if(i>0) {
        String str = str.concat(new.substring(0; i));
      }
      if(i<new.length-2) {
        str = str.concat(new.substring(i+2; new.length-2));
      }
      new= str;
      if(new.length == 2) ~~&& new.charAt(0) == new.charAt(~~
      {
        if(new.charAt(0)= new.charAt(i)) {
          return true;
        }
        else {
          return false;
        }
  }
```

String vs. stack-?

Compare opening braces with closing counterparts

5

Use the backside, if needed

DS.MT1.181016.M156