Name and ID#:

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

6/21

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

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.

```
int n ; function?
cin >> n;
if ( n == 1){
 return 1
 }
if (n > 1){
int b[n] ; init?
if (n > 1 ; b[n], n-- ){
b[n] = 2 * b[n-1] - b[n-2] * b(n-1)
if (n > 1 ; n-- ) {
return a_{n-1} * b_{n-1} ; ?
```

or

```
int n;
if ( n == 1){
 return (1) }
if ( n > 1) { int i ;
return b[i] = 2* b[i-1] - b[i-1] * b[i-1]
? return a[i] = a[i-1] * b[i-1]
```

2

DS.MT1.181016.L16f

Name and ID#:

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.

```
bool balanced_brackets(string text){
stack<char> letters;
string a;
char c< a;
int i,i1,i2;
[illegible]
for(i=0; t<text.letter(); i++) {
if text{i} =='(' || =='{' || =='[' || =='<'}
  letters.char(text(i))
for(int i=0; i<text.letter(); i++){
if text{i} =='(' || =='}' || =='}' || =='>'}}
if vtext{i1} < vtext{i2}
  letters.char  letters.char
cout<<" text is balanced"
else{
  cout<<" text is not balanced"
}
}
```

1

Use the backside, if needed