**Name and ID#:**__________

**AMERICAN UNIVERSITY OF ARMENIA**

*College of Science and Engineering*

**CS 121 Data Structures and Algorithms**

## MIDTERM 1 EXAM

8/21

| | |
|---|---|
| **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:

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

$b_3 = 1 + 2b_1 + 2b_2 - b_2 b_2$

$b_4 = 1 + 2 + 4 - 9$

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 $a_n$.

for bn

| | | |
|---|---|---|
| n=1 | 1 | 1 |
| n=2 | 2 | $1 + 2b_1 - b_1 \cdot b_1$ |
| n=3 | 3 | $1 + 2b_1 + 2b_2 - b_2 \cdot b_2$ |
| n=4 | -2 | $1 + 2b_1 + 2b_2 + 2b_3 - b_3 b_3$ |
| n=5 | | |

Henolint n-1)·

Heno(int n-1)

```
int    Heno(int n){
    if(n == 1)
     return 1;
    else {
 int a = 2 Henol int n-1);
 int b = Heno(int n-1)·Heno(int n-1)
 ~~return~~
 return 1+ (a - b)· Heno(n-1);

           }
```

3

*Use the backside, if needed*

DS.MT1.181016.H163

**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.

~~for(int i=0;~~

~~{ int a=0, b=0, c=0, d=0~~

~~for(int i=0; i< .length; i++){~~

~~cin>>~~ p;

So as the number of each type of bracket is important we need to declare 4 types of char?

```
stack <char> a, b, c, d
for( int i=0: p.length) {
if( p[i] = '(' or P[i]= ')')
        a. push<stack>;
if (P[i]='[' or P[i] = ']' )
        b. push <stack>;
if (p[i]= '{' or p[i]='}')
        c. push <stack>;
if (p[i]= '<' or p[i]= '>')
        d. push <stack>;
If (a is even && b is even && c is even)
                             && d is even)
return true;
else
  return false;
```

number of elements in stack with a; number of element b; number of element c

Use the backside, if needed

3

DS.MT1.181016.H163