UNCA CSCI 202
Exam 2 Spring 2012
11 April, 2012

This is a closed book and closed notes exam. This is a 90-minute exam to be given from 8:20 AM to 9:50 AM. Calculators, PDA's, cell phones, and any other electronic or communication devices may not be used during this exam.

Name: _____

If you want partial credit for imperfect answers, explain the reason for your answer!

Problem 1 (10 points)
Complete the `pushMe` method so that it pushes your name 2012 times on the `Deque<String>` passed to `pushMe`.

```
public static void pushMe(Deque<String> dq) {




}
```

Problem 2 (9 points)
Suppose there is a class `Match4Exam` with methods with the following headers:

```
public        int    niceOpA (int i, double y) ;
public        double niceOpB (String[] s) ;
public static double niceOpC () ;
```

and that `niceX` and `niceY` are objects of type `Match4Exam`.

Write Java statements to do each of the following:

Invoke `niceOpA`
Invoke `niceOpB`
Invoke `niceOpC`

Problem 3 (21 points)
The table below contains two columns. The leftmost column is a Java expression. In the rightmost column write the value of the expression as a Java literal. If the value is a `double`, `boolean`, or `String` be sure to use the Java syntax for writing `double`s, `boolean`s, or `String`s. Some of these are tricky, but none require complex arithmetic.

`(int) 3.14`	
`5 * (2 / 5)`	
`(5 * 2) / 5`	
`5 + 0.0`	
`1.0 / 2 / 1`	
`1 / 2 / 1.0`	
`(double) 3 / 2`	
`true == false`	
`true || false`	
`true && false`	
`("" + 5) + 6`	
`"" + (5 + 6)`	
`"cat".length()`	
`"cat".indexOf('a')`	

Problem 4 (12 points)
Suppose `V` is an array of `String`s (`String[]`). Write a short section of Java code that declares and creates a new array of `String`s called `revV` which contains the same elements as `V`, but in reversed order. Your program should work even if `V` has no elements.

Write your answers to these problems on lined paper.

Problem 5 (30 points)
There are five parts to this question. Each is worth six points.

Declare `sub5A` to be an array of `String`s, then initialize `sub5A` so that it contains 100 `String`s all having the identical value `"Same Thing"`.

Declare `sub5B` to be a `Deque` of `String`s, then initialize `sub5B` so that it contains 100 `String`s all having the identical value `"Same Thing"`.

Assume that `sub5C` is an array of `String`s. Write a section of code that sets the last element of `sub5C` to `"New Thing"` when `sub5C` contains at least one element.

Assume that `sub5D` is an `Deque` of `String`s. Write a section of code that sets the last element of `sub5D` to `"New Thing"` when `sub5D` contains at least one element.

Assume that `sub5E` is a `Deque` of `String`s. Write a section of code to add up the length of ***all*** the `String`s of `sub5E` and store the result in an integer variable `total5E`. Full credit will be given only if your code does *not* modify `sub5E`.

Problem 6 (18 points)
In this problem assume that `matchUp` is a `String` that contains only parentheses `()` or brackets `[]`. We consider `matchUp` balanced if the right and left parentheses and the right and left brackets are properly matched even though they may be nested. For example, the following three strings are balanced:

`"()()[][]"`	No overlap at all
`"[([])[()]]"`	Proper nesting
`""`	Empty is considered acceptable

However, all of the following are unbalanced:

`"()(][]"`	One unmatched pair
`"(()"`	Too many lefties
`"())"`	Too many righties

Write a section of Java code that tests if `matchUp` is balanced and prints `OK` if it is.

I strongly suggest that you use a `Deque<Character>`, with its `push` and `pop` operations, to solve this problem. `push` on lefties and `pop` on righties, and make sure every righty gets matched with the appropriate lefty. Explain your answer for partial credit.

Thing to remember:

1) When the `Deque` is empty, the `size()` method returns 0.
2) When the `Deque` is empty, the `pop()` method returns `null`.
3) The numbers of characters in `matchUp` is `matchUp.length()`.
4) The *i*'th character of `matchUp` is `matchUp.charAt(`*i*`)`.
5) Liberal partial credit is given for code with the "right idea", even if the Java isn't perfect.