Exam Number:

Points off	1	2	3	4	5	6		Total off	Net Score

CS 305j – Midterm 2 – Fall 2006

Your Name___________________________________

Your UTEID _________________________________

Circle you TA's name: Brad Jacob

Instructions:

1. Please turn off your cell phones
2. There are 6 questions on this test.
3. You have 50 minutes to complete the test.
4. You may not use a calculator.
5. Please make your answers legible.
6. When code is required, write Java code.

1. Simulation. (3 points each, 12 points total.) You are to simulate a method that manipulates an array of integers. Consider the following method:

```
public static void mystery1(int[] data){
     for(int index = 1; index < data.length; index++){
          data[index] = data[index - 1] + data[index];
     }
}
```

In the left hand column below is an indication of the values initially in array of integers. The left most element is at position 0. In the right hand column indicate what the elements of the array would be after method `mystery1` finished executing when the array in the left hand column is passed as the argument to method `mystery1`.

Initial Array	Resulting Array after method `mystery1`
`{12}`	______________________________
`{3, 5}`	______________________________
`{6, 1, 5}`	______________________________
`{-2, 1, -2, 2}`	______________________________

2. Arrays I. (20 points) Complete a method `averageOfValuesInRange`. This method has three parameters, an array if integers, a lower limit, and an upper limit. The array calculates and returns the integer average of all elements of the array that are between the lower limit and upper limit inclusive. You may assume there will be at least one element in the array that is in range.

The method you complete will have the following header:

```
public static int averageOfValuesInRange(int[] data, int lower, int upper)
```

Here are some examples of call to `averageOfValuesInRange` and what value should be returned:

```
averageOfValuesInRange( {1, 2, 3}, 1, 3 ) -> 2
averageOfValuesInRange( {1, 2, 3, 4}, 2, 3 ) -> 2
averageOfValuesInRange( {5, 1, 3, 7}, 4, 10 ) -> 6
```

Complete the method below:

```
// lower <= upper
// there will be at least one element of data that is in range
public static int averageOfValuesInRange(int[] data, int lower, int upper){
```

3. Boolean expressions. (20 points) Complete a method that determines if the next opponent for the UT women's volleyball team is a difficult opponent. An opponent is difficult based on the opponent's winning percentage and whether the match is to be played at UT or away from UT. Win percentage is the number of matches the opponent has won divided by the total number of matches the opponent has played. Total number of matches played is equal to the number of matches won plus the number of matches lost. If playing at UT opponents with a win percentage greater than or equal to .85 are difficult opponents. If playing away from UT opponents with a win percentage greater than or equal to .75 are difficult opponents.

The method you complete will have the following header:

```
public static boolean toughOpponent(int wins, int losses, boolean atUT)
// The parameter wins equals the number of matches the opponent has won.
// The parameter losses equals the number of matches the opponent has lost.
```

Here are some examples of call to `toughOpponent` and what value should be returned:

```
toughOpponent(20, 0, false) -> true     // winning percentage = 1.0
toughOpponent(8, 2, true) -> false      // winning percentage = 0.8
toughOpponent(8, 2, false) -> true      // winning percentage = 0.8
toughOpponent(5, 5, false) -> false     // winning percentage = 0.5
```

Complete the method below:

```
public static boolean toughOpponent(int wins, int losses, boolean atUT) {
```

4. Strings. (20 points) Complete a method named `triple_abi`. Method `triple_abi` takes in one parameter, a String. The method returns a String. The returned String has the following properties. Each 'a', 'b', and 'i' in the original String result in three occurrences of that character in the resulting String. Thus 'a' in the original String will appear as "aaa" in the resulting String. Only lower case 'a', 'b', and 'i' are tripled in this way. All other characters in the original String, except dashes, are doubled in the resulting String. Thus 'M' in the original String would become "MM" in the resulting String. Dashes in the original String do not appear in the resulting String. You will need the `length()` and `charAt(int pos)` methods from the String class. Recall that character positioning in Strings, like arrays, starts at 0, not 1.

The method you complete will have the following header:

```
public static String triple_abi(String org)
```

Here are some examples of call to `triple_abi` and what value should be returned:

```
triple_abi("a") -> "aaa"
triple_abi("A") -> "AA"
triple_abi("A-a") -> "AAaaa"
triple_abi("isabelle") -> "iiissaaabbbeellllee"
triple_abi("") -> ""
triple_abi("a-B-B-a") -> "aaaBBBBaaa"
```

Complete the method below:

```
public static String triple_abi(String org){
```

```
// more room on next page if necessary
// more room for method triple_abi
```

5. Analysis of method specification. (6 points). Explain how could method `triple_abi` could be made more general. Include a method header for a more general version.

6. Arrays II (22 points) Complete a method `moveTargetToBack`. This method has two parameters, an array of integers and an integer that is the target value. All elements of the array that are equal to the target value are moved to the end of the array. All elements not equal to the target value are moved to the front of the array and maintain their relative position.

The method you complete will have the following header:

```
public static void moveTargetToBack(int[] data, int tgt)
```

Here is an example of an initial array.

```
{12, 1, 3, -5, 1, 1, 7, 1, 15}
```

If `tgt` were equal to 1, the elements of the array would be rearranged as follows:

```
{12, 3, -5, 7, 15, 1, 1, 1, 1}
```

Notice the four 1s, which are equal to `tgt`, have been moved to the back of the array and the elements not equal to 1 have been shifted forward, but have maintained their relative order.

Complete the method below:

```
public static void moveTargetToBack(int[] data, int tgt){
```

```
// more room on next page if necessary
// more room for method moveTargetToBack
```

Scratch Paper

Scratch Paper

Scratch Paper

Scratch Paper

Scratch Paper