Midterm 2

CS 1323, Fall 2017

Name (printed legibly):

Student number:

Integrity Pledge: On my honor, I affirm that I have neither given nor received inappropriate aid in the completion of this exercise.

Signature:

Do not write anything on the back of any page.

If you need additional space, use the blank pages at the end of the examination. If you pull your examination apart, put ***all of your examination pages in order*** when you turn them in and turn in all pages.

Answer all programming questions in Java.

Pay careful attention to what is requested: a code fragment (a few lines of code), a method, or a program. Students who write a whole program when a code fragment is requested will waste so much time that they may not complete the examination.

Unless otherwise indicated, each part of a problem is worth the same number of points. Show your work to receive partial credit.

When a variable has a comment like "value assigned elsewhere," it means that the value of the variable has already been created somewhere out of sight (like in Turingscraft). You do not need to prompt the user or get values from the user in this situation.

Since you do not have the whole API available during the examination, it is acceptable to guess at method names, parameters and return values. If your guesses are reasonable—even if not perfect—you will receive credit. For example, if you forget that the String class has a length() method and call it size(), that is fine. If, however, you make up new methods that are not reasonable for the class or that magically solve problems the class cannot solve, you will not get credit.

You do not need to import packages or throw exceptions in any methods.

1. (12 points; 4 points each) Find the logical value (true or false) of each of the statements below. If the code istru not legal, say so. Assume the variables are declared and initialized as follows:

```
double ghosts = 9.3 ;
double goblins = 7.1 ;
int[] candy = {0, 5, 1, 2, 1, 4};
```

a) candy[3] > 2 || candy[5] != 5
   2 > 2 || 4 != 5
   false || true
   true

b) !(ghosts == goblins)
   ! (9.3 == 7.1)
   ! false
   true

d) candy[0] != candy[1] || candy[1] >= candy[2] && candy[2] > candy[3]
   0 != 5 || 5 >= 1 && 1 > 2
   true || false && true
   true || false
   true

2. (6 points) I'm buying Halloween candy to give to kids that live near me. Usually I buy two pieces of candy for each trickOrTreater I expect. However, I've noticed that if it doesn't rain and the temperature is above 60 that I need to twice as much candy. ***Use all of the variables declared below and logical operator(s) to write a single if/else statement*** that will calculate how much candy I should buy. Pay attention to precedence of the logic operators.

```
int trickOrTreaters; // value given elsewhere
boolean isRaining; // value given elsewhere
int temperature; // value given elsewhere
int candy; // this is the value you should set

if (!isRaining && temperature > 60) // isRaining==false works too, as does !(isRaining==true)
{
        candy = 4 * trickOrTreaters;
}
else
{
        candy = 2 * trickOrTreaters;
}
```

Another shorter solution is:

```
candy = 2 * trickOrTreaters;
if (!isRaining && temperature > 60)
{
        candy = candy * 2;
}
```

3. (15 points; 5 points for table on right, 5 points for table below, 5 points for the final values in the array) Trace the loop below in the tables. Remember to show initial and final values.

```
String[] names = {"Tiana", "Ariel", "Miridia", "ariel", "Elsa", "Jasmine"};
for (int index = 0; index < names.length-1; ++index)
{
        if (!names[index].equalsIgnoreCase(names[index+1]))
        {
                String temp = names[index+1];
                names[index+1] = names[index];
                names[index] = temp;
        }
}
```

index	temp
0	Ariel
1	Miridia
2	ariel
3	Elsa
4	Jasmine

names[-1]	names[0]	names[1]	names[2]	names[3]	names[4]	names[5]
	Tiana	Ariel	Miridia	ariel	Elsa	Jasmine
	Ariel	Tiana				
		Miridia	Tiana			
			ariel	Tiana		
				Elsa	Tiana	
					Jasmine	Tiana

At the end of the method, names contains: {"Ariel", "Miridia", "ariel", "Elsa", "Jasmine", "Tiana"}

4. (15 points; 4 points for a), 6 points for b); 5 points for c)) The paragraph below describes a static method.

The method will take an array of doubles and replace any values in the array that are larger than a given target value by 0.0.

For example, if the array contained {7.9, 2.3, 4.1, 5.2, 6.3}, and the target value were 6.0, the array would contain {0.0, 2.3, 4.1, 5.2, 0.0} after the method call.

a) Write the signature of the method. The method signature contains a return type, the method name and any necessary parameter(s).

void biggerValues(int[] source, int target)

b) Complete the declaration of variables below and call the method with the data given in the example above.

```
double[] array = {7.9, 2.3, 4.1, 5.2, 6.3};
double target = 6.0;
biggerValues(array, target);
```

c) What will the value of index be after the code below is run?

```
int[] array = {9, 4, 2, 1, 7, 3, 5};
Arrays.sort(array);
int index = Arrays.binarySearch(array, 6);
```

After sort, array contains: 1, 2, 3, 4, 5, 7, 9

Six would be inserted where 7 is at index 5.

The method therefore returns -5-1 = -6

5. (12 points) Answer the questions in the box below about program execution. You may use a memory diagram to trace the code if you wish to, or may use your knowledge of passing by value and/or sharing.

```
public class SecondMidterm {
	public static void main(String[] args) {
		int[] start = {1, 7, 5};
		int value = 3;
		start = mystery(value);
	}

	public static int[] mystery(int sub) {
		sub = sub + 1;
		int[] data = new int[sub];

		for (int index=0; index<sub; ++index) {
			data[index] = sub * index;
		}
		return data;
	}
}
```

At the end of the program, the variables contain:

start (8 points): {0, 4, 8, 12} or 1004

value (4 points): 3

main stack frame

Identifier	Address	Contents
start	100	~~1000~~ 1004
value	101	3
	102	
	103	

mystery stack frame

Identifier	Address	Contents
sub	200	~~3~~ 4
data	201	1004
	202	
	203	
	204	

heap

Identifier	Address	Contents
0	1000	1
1	1001	7
2	1002	5
length	1003	3
0	1004	0
1	1005	~~0~~ 4
2	1006	~~0~~ 8
3	1007	~~0~~ 12
length	1008	4
	1009	
	1010	
	1011	
	1012	

6. (40 points; 10 points for a), 10 points for b), 20 points for c)) There are many automated software products that implement surveys (SurveyMonkey, Qualitrics, etc.). Most of these products store the responses to a survey in a file so you can analyze the survey yourself.

I've given a survey as follows:

Computer science is the greatest discipline on earth. Choose a response from below:
Agree
Neutral
Disagree
Not applicable: <fill in the reason here>

I've stored the responses in a file.

Write a program that calculates the number of Agree, Neutral, and Disagree responses and lets a user explore which Not applicable: responses contain a given word or words until the user wishes to quit. The expected interaction of the program is shown below. The sample file contents are in the box on the right. The bold, underlined, italicized words were entered by the user

```
Agree: 6
Neutral: 1
Disagree: 5
Enter a keyword to search not applicable or quit
Uncertain
That keyword was listed 2 times
Enter a keyword to search not applicable or quit
not sure
That keyword was listed 0 times
Enter a keyword to search not applicable or quit
quit
```

```
Agree
Agree
Agree
Disagree
Disagree
Disagree
Neutral
Not applicable: Uncertain
Agree
Agree
Not applicable: Who knows?
Not applicable: Uncertain
Not applicable: Don't care
Not applicable:
Not applicable:
Agree
Disagree
Disagree
```

You will write the two methods below and part of the main method.
public static int countOtherKeywords(String[] data, int dataSize, String answer)
public static int readFile(String[] data, String fileName)

There also is another method that you may use without writing:
public static int countResponses(String[] data, int dataSize, String answer)
This method returns the number of times that answer appears in the data array at indices between 0 and dataSize-1 (inclusive).

a) Write the method described below.

public static int readFile(String[] data, String fileName)

Read survey responses stored in a file with the given fileName into an array. Returns the number of elements in the array when the method is complete. This should equal the number of lines in the file.

```
public static int readFile(String[] data, String fileName) throws
FileNotFoundException
{
	Scanner file = new Scanner(new File(fileName));

	int index = 0;
	while (file.hasNextLine() && index < data.length)
	{
		data[index] = file.nextLine();
		++index;
	}

	file.close();
	return index;

}
```

b) Write the method below.

public static int countOtherKeywords(String[] data, int dataSize, String answer)

Return the number of times Not applicable: <something containing answer> occurs in the elements of data. For example, if answer were "don't", both "Not applicable: don't know" and "Not applicable: don't care" would be counted. “Not applicable: Don’t know” would not be counted (because D and d don’t match).

The String class has a method called boolean contains(String source) that determines whether this String does or does not contain source. For example, if the String test contains “hello”, test.contains(“lo”) will return true and test.contains(“jello”) will return false.

```
public static int countOtherKeywords(String[] data, int dataSize,
                  String answer)
{
      int count = 0;
      for(int index = 0; index < dataSize; ++index)
      {
            if (data[index].startsWith("Not applicable:")
                        && data[index].contains(answer))
            {
                  ++count;
            }
      }

      return count;
}
```

c) Complete the boxed areas of the main program below. You must call the methods you wrote in a) and b). (4 points for first box, 4 points for second box, 6 points for third box, 6 points for fourth box)

```
public static void main(String[] args) throws FileNotFoundException
      {     // set up the Scanner
            Scanner keyboard = new Scanner(System.in);

            // Set up the array.
```

```
              String[] survey = new String[SIZE];
              int surveySize = 0;
```

```
            // Read in data from the file named SurveyResponses.txt
```

```
              surveySize = readFile(survey, "SurveyResponses.txt");
```

```
            // Show the number of Agree, Neutral and Disagree responses
```

```
              int count = countResponses(survey, surveySize, "Agree");
              System.out.println("Agree:" + count);
              count = countResponses(survey, surveySize, "Neutral");
              System.out.println("Neutral:" + count);
              count = countResponses(survey, surveySize, "Disagree");
              System.out.println("Disagree:" + count);
```

```
            // Let the user analyze not applicable responses if they wish to
            // Priming read
            System.out.println("Enter a keyword to search not applicable or"
                  + "Quit to end");
            String response = keyboard.nextLine();
            while(!response.equalsIgnoreCase("quit"))
            {     // Search the array for the user's requested response
```

```
                  count = countOtherKeywords(survey, surveySize, response);

                  System.out.println("That keyword was listed under other "
                              + count + " times");
```

```
                  //Priming read
                  System.out.println("Enter a keyword to search not "
                        + "applicable or quit to end");
                  response = keyboard.nextLine();

            }
            keyboard.close();
      }
```

Do not write anything on the back of any page