Midterm 2

CS 1323, Fall 2016

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.

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. (9 points; 3 points each) Find the logical value (true or false) of each of the statements below. If the code is not legal, say so. Assume the variables are declared and initialized as follows:

```
int tricks = 19;
double treats = 42.3;
int[] candy = {8, 2, 4, 1 3, 7, 5};
```

a) tricks < 30 && treats > 50
   19 < 30 && 42.3 > 50
   true && false
   false

b) candy[0] == 8 || candy[0] < candy[1]
   8 == 8 || 8 < 2
   true || false
   true

c) ! (candy[2] < candy[4])
   ! (4 < 3))
   ! false
   true

2. (6 points) I'm considering getting a new dog. I have very strict requirements. All of my dogs have been black, and I want to continue to have black dogs. I also like dogs that are either more than 50 pounds or between five and fifteen pounds. All weights are inclusive. If the dog's color is stored in the String variable color below and the dog's weight is stored in the double variable weight, write a single if statement that will print out "I'd like to meet you" if an adoptable dog meets my requirements.

```
String color; // value assigned elsewhere
double weight; // value assigned elsewhere

if (color.equalsIgnoreCase("Black") && (weight >= 50 || (weight >= 5 && weight =< 15)))
{
        System.out.println("I'd like to meet you.");
}
```

1. (15 points) Trace the loops below in the tables. Remember to show values at the start and the end.

a)

```
double[] array = {42.7, 31.5, 1.9, 27.2, 63.8, 52.9};
int sum = 0;
int threshold = 30;
for (int index = 0; index < array.length; ++index)
{
        if (array[index] > threshold)
        {
                threshold = threshold + 5;
                sum = sum + 1;
        }
}
```

sum	threshold	index	array[index]
0	30	0	42.7
1	35		
		1	31.5
		2	1.9
		3	27.2
2	40	4	63.8
3	45	5	52.9
		6	

b)

String[] files = {“methods.pptx”, “methods.docx”, “homework.txt”, “array.pptx”, “array.txt”, “logicoperators.pptx”, “methods.docx”};

```
int longest = 10000;
String result=”initial”;
// READ LOOP CAREFULLY
for (int index = files.length-1; index >=0; --
index)
{
        result = fileNames[index];
        if (result.length() < longest)
        {
                longest = result.length();
        }
}
```

longest	result	result.length	index
10000	initial		
11	methods.docx	11	6
	logicoperators.pptx	19	5
9	array.txt	9	4
	array.pptx	10	3
	homework.txt	12	2
	methods.docx	12	1
	methods.pptx	12	0
			-1

2. (20 points; 5 points each) The paragraphs below describe static methods. Determine the signature of the method **ONLY**. The signature of a method is the return type, the method name, and the parameters.

Write a method that takes two given arrays of int of the same size and subtracts the elements at matching indices putting the result in an array. Neither of the given arrays should be altered by the method.

For example: if the first array contained {1, 3, 5, 7, 9} and the second contained {9, 7, 5, 3, 1}, the result array would contain {-8, -4, 0, 4, 8}.

Do not write the method(s).

a) Assume all of the arrays are perfect size.

int[] subtract(int[] first, int[] second)

b) Assume all arrays are oversized.

int subtract(int[] first, int[] second, int arraySize, int[] result) // may have two sizes

c) Call the method with perfect sized arrays below. There may be some extra variables that are unnecessary for perfect size arrays.

```
int[] first = {1, 3, 5, 7, 9};
int firstSize = 5;
int[] second = {9, 7, 5, 3, 1};
int secondSize = 5;
int[] result;
int resultSize;

result = subtract(first, second);
```

d) Call the method with oversized arrays.

```
int[] first = {1, 3, 5, 7, 9, 0, 0, 0};
int firstSize = 5;
int[] second = {9, 7, 5, 3, 1, 0, 0, 0};
int secondSize = 5;
int[] result;
int resultSize;

result = new int[first.length];

resultSize = subtract(first, second, firstSize, result);
```

3. (10 points) Show the contents of the three data elements in the main program when the program finishes 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. ***Show or explain your work to get partial credit.***

```
public class MemoryDiagram {
	public static void main(String[] args) {
		int first = 9;
		double second = 4.5;
		char[] initials = {'a', 'c', 'd', 'f'};
		first = change(initials, second);
		}

	public static int change(char[] data, double bound)
	{
		data[0] = data[1];
		data[1] = data[2];
		bound = bound + 1;
		return (int) bound;
	}

}
```

```
At the end of the program, the variables contain:
first (2 points): 4
second contains (3 point): 4.5
initials contains (5 points): {'c', 'd', 'd', 'f'}
```

The tables below are for your use if you wish to trace the method to determine what is in array. You do not have to use these tables if you do not wish to.

main stack frame

Identifier	Address	Contents
first	100	~~9~~ 5
second	101	4.5
initials	102	1000
	103	

change stack frame

Identifier	Address	Contents
data	200	1000
bound	201	~~4.5~~ 5.5
	202	
	203	
	204	

Heap

Identifier	Address	Contents
0	1000	~~a~~ c
1	1001	~~c~~ d
2	1002	d
3	1003	f
length	1004	4
	1005	
	1006	
	1007	
	1008	
	1009	
	1010	
	1011	
	1012	

4. (40 points; 15 points for a), 10 points for b), 15 points for c))

I considering starting a business to make and sell Halloween costumes. I'm going to do some market research this Halloween at my house to figure out which costumes will be popular next year. To help myself out, I'm going to write a program that uses an oversized array to keep track of short descriptions of costumes of kids are wearing when they visit my house this year. The program will take user input to describe each costume and store these descriptions in an oversized array. Then it will allow the user to enter words and find out how many costumes met that criteria.

You may assume that I never get more than 200 kids Trick or Treating at my home. Your program may use only one Scanner object. A typical run of the program is shown below.

```
Describe the next child's costume or Quit to exit
Bozo the clown
Describe the next child's costume or Quit to exit
Miridia in green dress
Describe the next child's costume or Quit to exit
Chuckles the clown
Describe the next child's costume or Quit to exit
Miridia in blue dress
Describe the next child's costume or Quit to exit
Grim reeper
Describe the next child's costume or Quit to exit
clown
Describe the next child's costume or Quit to exit
reeper
Describe the next child's costume or Quit to exit
quit
Enter the first description to search for
clown
Enter the second description to search for
reeper
There were 5 costumes that contained clown or reeper
```

Do not write code on this page or on the back of any page.

You must write the methods exactly as they are given. You may not change the methods, parameters, or return type. You must use the methods you write in a) and b) in the main program in c).

a) int recordCostumes(String[] costumes, String sentinel, Scanner keyboard, String prompt). This method prompts the user and allows the user to enter costume descriptions until the sentinel (“Quit” in the example above) is found. The int returned is the number of costumes entered in the array.

public static int recordCostumes(String[] costumes, String, sentinel, Scanner keyboard, String prompt)

```
public static int recordCostumes(String[] descriptions, String sentinel,
Scanner keyboard, String prompt)
{
	int count = 0;
	System.out.println(prompt);
	String input = keyboard.nextLine();
	while (!input.equalsIgnoreCase(sentinel))
	{
		descriptions[count] = input;
		++count;

		System.out.println(prompt);
		input = keyboard.nextLine();

	}

	return count;
}
```

b) int countInEitherDescription(String[] costumes, int costumeSize, String key1, String key2). The method searches the array for any description that contains either key1 or key2. The number of times a description that has the required key(s) is returned.

There is a useful method in the String class: contains(String target). This method determines whether one String contains the target. For example:
String source = “Hello world”;
source.contains(“lo”) returns true;
source.contains(“Goodbye”) returns false;

```
public static int countInEitherDescription(String[] descriptions, int
descriptionSize, String key1, String key2)
{
	int count = 0;
	for(int index = 0; index<descriptionSize; ++index)
	{
		if (descriptions[index].contains(key1) ||
			descriptions[index].contains(key2))
		{
			++count;
		}
	}
	return count;
}
```

c) Write the main program using the methods in a) and b).

The method signatures from a) and b) are:
int countInEitherDescription(String[] costumes, int costumeSize, String key1, String key2)
int recordCostumes(String[] costumes, String sentinel, Scanner keyboard, String prompt)

```
public static void main(String[] args)
{
	Scanner keyboard = new Scanner (System.in);
	final int SIZE = 200;
	String[] costumes = new String[SIZE];
	int costumesSize = 0;

	costumesSize = recordCostumes(costumes, "quit", keyboard,
			"Describe the next child's costume or Quit to exit");

	String target1;
	String target2;
	System.out.println("Enter the first description to search for ");
	target1 = keyboard.nextLine();
	System.out.println("Enter the second description to search for ");
	target2 = keyboard.nextLine();

	int count = countInEitherDescription(costumes, costumesSize,
			target1, target2);
	System.out.println("There were " + count + " costumes that contained "
		+ target1 + " or " + target2);

	keyboard.close();
}
```