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

    b) candy[0] == 8 || candy[0] < candy[1]

    c) ! (candy[2] < candy[4])

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

2. (15 points) Trace the loops below in the tables.

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]

b)

String[] files = {"methods.ppt", "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 = files[index];
        if (result.length() > longest)
        {
                longest = result.length();
        }
}
```

longest	result	result.length	index

3. (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.

b) Assume all arrays are oversized.

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;

d) Call the method with oversized arrays. There may be some extra variables that are unnecessary for 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;

4. (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.***

```java
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):
second contains (3 point):
initials contains (5 points):
```

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, although it is one way to gain partial credit.

main stack frame

Identifier	Address	Contents
	100	
	101	
	102	
	103	

Heap

Identifier	Address	Contents
	1000	
	1001	
	1002	
	1003	
	1004	
	1005	
	1006	
	1007	
	1008	
	1009	
	1010	
	1011	
	1012	

change stack frame

Identifier	Address	Contents
	200	
	201	
	202	
	203	
	204	

5. (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 that describes each costume, stopping when a sentinel is entered, and store these descriptions in an oversized array. Then it will allow the user to enter words and find out how many costumes meet that criteria. Your program should be case sensitive; in other words, "Clown" and "clown" are different words.

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) This method prompts users 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)

b) The method searches the array for any description that contains either target1 or target2. 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[] costumes, int costumeSize,
                              String target1, String target2)

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

The method signatures from a) and b) are:
public static int countInEitherDescription(String[] costumes, int costumeSize,
        String target1, String target2)
public static int recordCostumes(String[] costumes, String sentinel, Scanner keyboard, String prompt)