Midterm 2

CS 1323, Spring 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.

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; 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 leprechauns = 3 ;
int shamrocks = 22 ;
double[] hoursWorked = {4.3, 0.0, 5.2, 8.1, 7.4};
```

a) leprechauns > 0 || shamrocks > 10

b) hoursWorked[1] > 0.5 && hoursWorked[5] < 8.0

c) ! hoursWorked[0] > 4.0

d) hoursWorked[0] != hoursWorked[1] && hoursWorked[1] < hoursWorked[2]
|| hoursWorked[2] > hoursWorked[3]

2. (6 points) I'm considering purchasing a car. I either want a two door car which gets more than 30 miles to the gallon, or a four door car that gets more than 20 miles to the gallon. Use the variables declared below ***to write a single if statement*** that will print out "This car fits your requirements" if a car meets my requirements. Pay attention to precedence of the logic operators.

```
int doors; // value given elsewhere
double milesPerGallon; // value given elsewhere
String prompt = "This car fits your requirements";
```

3. (15 points; 6 points for a), 9 points for b)) Trace the loops below in the tables. Remember to show initial and final values.

a)

```
int[] array = {1, 3, 5, 7, 8, 9};
int count = 0;
int target = 6;
for (int index = 0; index < array.length && array[index] < target;
                ++index)
{
        ++count;
}
```

index	count	array[index]

b) (6 points for table, 3 points for box)

```
String[] names = {"Shrek", "Donkey", "Fiona", "Duloc", "Dragon", "Farquaad"};
for (int index = 0; index < names.length/2; ++index)
{
        String temp = names[index];
        int mirror = names.length – index – 1;
        names[index] = names[mirror];
        names[mirror] = temp;
}
```

index	temp	mirror

At the end of the method, names contains:

Do not write anything on the back of any page

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 integers and create a new array that contains only values that are bigger than a given target value in their original order.

For example, if the original array contained {9, 4, 2, 1}, and the target value were 3, the new array would contain {9, 4}.

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

b) Call the method with the data given in the example above. Use the variables below. You may not declare any other variables. The variable array may not be initialized using {} notation.

```
int[] array;  // you may not change this line
int target;
int[] result; // the result array should go here
```

c) The Arrays class could (but doesn't) contain the method described below.

static int indexOfSubList(int[] source, int[] target)

Returns the starting position of the first occurrence of the specified target array elements within the specified source array or -1 if there is no such occurrence. For example, if source contained {1, 3, 5, 7} and target contained {3, 5} the method would return 1. If target contained (4, 6}, the method would return -1. Use the method to find if two arrays (first and second, below) contain the exact same values. If the values are the same print out "Same". If the values are different, print out "Different".

```
int[] first; // Array constructed and assigned values elsewhere
int[] second; // Array constructed and assigned values elsewhere
```

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 MemoryDiagram {
	public static void main(String[] args) {
		int length = 5;
		int contents = 2;
		int[] array = {3, 2, 1};
		passMe(length, contents, array);
		}

	public static int[] passMe(int size, int value, int[] source)
	{
		int[] result = new int[size];
		for (int index = value; index > 0; --index)
		{
			result[index] = value;
			source[index] = value;

		}
		return result;

	}
}
```

```
The method passMe returns (4 points):
At the end of the program, the variables contain:
	length(2 points):
	content(2 points):
	array (4 points):
```

main stack frame

Identifier	Address	Contents
	100	
	101	
	102	
	103	

passMe stack frame

Identifier	Address	Contents
	200	
	201	
	202	
	203	
	204	

heap

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

6. (40 points; 10 points for a), 15 points for b), 15 points for c)) You have been asked to write a program that will analyze retail theft for a unique store that sells only one item. You will be given the starting inventory, the daily sales for a week, and the closing inventory. If there were no theft, the starting inventory minus the daily sales would equal the closing inventory. If the closing inventory is less than expected, it is likely there was theft.

The expected interaction of the program is shown below. The numbers were all entered by the user.

```
Enter the starting inventory
100
Enter day 1 sales.
5
Enter day 2 sales.
7
Enter day 3 sales.
3
Enter day 4 sales.
4
Enter day 5 sales.
6
Enter day 6 sales.
10
Enter day 7 sales.
3
Enter the ending inventory
23
There was theft
```

If the ending inventory matches the calculated inventory, print out “There was no theft.” If the ending inventory is smaller than the calculated inventory, print out “The inventory is in error.”

You must use an array to write this program, and use the exact method signatures given.

Write the methods on the next pages.

a) Write the method below. The method will sum the values in sales, and return the sum.

If sales contained {1, 2, 3, 4, 5} then 15 would be returned. Do not assume that the sales array has any specific number of elements.

public static int sum(int[] sales)

b) Write the method below. The method takes a Scanner object and a number of days and allows the user to enter one sales number for each day. If days was 3, the user could enter 5, 6, 7 and the method should return an array with three elements {5, 6, 7}.

public static int[] getDailySales(Scanner input, int days)

c) Write the main program. You must call the methods you wrote in a) and b). The method signatures are below:

int sum (int[] sales)

int[] getDailySales(Scanner input, int days)