CSE 131 Introduction to Computer Science **Fall 2016**

Exam I

Given: 29 September 2016 Due: End of Exam Session

This exam is closed-book, closed-notes, no electronic devices allowed. The exception is the "sage page" on which you may have notes to consult during the exam. Answer questions on the pages of the exam. Do not unstaple the pages of this exam nor should you attach any other pages to the exam. You are welcome to use the blank space of the exam for any scratch work.

Your work must be legible. Work that is difficult to read will receive no credit. Be sure code is indented to show its structure. Use Comments if you think you need to clarify your work.

You must sign the pledge below for your exam to count. Any cheating will cause the students involved to receive an F for this course. Other action may be taken. If you need to leave the room for any reason prior to turning in your exam, you must leave your exam and any electronic devices with a proctor.

YOU MUST FILL IN YOUR IDENTIFYING INFORMATION CORRECTLY. Failure to do so is grounds for a zero on this exam.

Print clearly the following information:
Name (print clearly):
Student 6-digit ID (print *really* clearly):
Your answers below tell us where to return your graded exam.
What time do you actually attend studio/lab? 11:30am 1pm 2:30pm 4pm
What room? 222 , 218, 216, or 214

Problem Number	Possible Points	Received Points
1	10	
2	10	
3	15	
4	35	
5	30	
Total	100	

Pledge: On my honor. I have neither given nor received any unauthorized aid on this exam.

Signed:

1. (10 points) Circle the correct type for each expression in the table below, and state the reult of evaluating the expression:

Expression	Type	Result
`(int)5.2 * 10`	double int boolean String	
`(3-(2-1)) == (3-2-1)`	double int boolean String	
`false || !((1 > 2) && true)`	double int boolean String	
`3.0/2`	double int boolean String	
`"(2*2.0)" + 1`	double int boolean String	
`(double)(5/10)*10`	double int boolean String	
`(int)((3.0/6)*8)`	double int boolean String	
`((double)1/2) + "+"`	double int boolean String	
`!false && !(false && !true)`	double int boolean String	
`(2/3) <= (1/2)`	double int boolean String	

2. (10 points) Complete the code below so that it prints true if exactly two of the given integers are equal, otherwise it prints false.

```
int a = ap.nextInt("Value for a?");
int b = ap.nextInt("Value for b?");
int c = ap.nextInt("Value for c?");
```

3. (15 points) Write code that will print out all of the integers from 1 to 200. If the integer value is divisible by 3, instead of printing out the integer, print out the word “pop”. If the integer value is greater than 90, instead of printing out the integer, print out the word “corn”. If the integer value is divisible by three *and* greater than 90, print out “popcorn”. An example of the output follows:

```
1
2
Pop
4
5
Pop
.
.
.
88
89
Pop
Corn
Corn
Popcorn
Corn
Corn
Popcorn
.
.
.
```

4. (35 points) You are given the following code:

```
int X = ap.nextInt("Value for X?");
double[][] arr = new double[X][X];
```

(a) (8 points) Write code that will fill the array with random values.

(b) (3 points) We wish to create an array that contains the sum of each column of `arr`. Write a single line of code that will create an empty array with enough space to store these values:

(c) (16 points) Go through the `arr` array, compute the sum of each column, and store the resulting sum in the array you created in the previous part. For example, if our random array looks like:

```
0.1 0.2 0.3
0.4 0.0 0.1
0.2 0.1 0.7
```

Then the array you created should contain:

```
0.7 0.3 1.1
```

(d) (8 points) Write code that will compute the sum of the diagonal of the array, from top left to bottom right, and print out the result.

5. (30 points) You work at a candy store that sells **C** different kinds of candy from **M** different candy makers. You need to calculate how much it is going to cost to order supplies for this month. You are given 3 arrays. One array, **pieces**, represents the number of pieces to order for each of the **C** types of candy. The second array, **makers**, represents the maker of each of the **C** types of candy. The third array, **prices**, represents the price per piece of candy that each of the **M** makers charge.

For example if **C** = 4 and **M** = 2 :

The **pieces** array might be 3 8 6 20 where the order is for 3 pieces of the first kind of candy and 20 of the fourth type.

The **makers** array might be 0 0 1 0 where the maker of the first kind of candy is maker 0 and the maker of the third kind of candy is maker 1.

The **prices** array might be 1.00 0.50 where maker 0 charges $1.00 per piece of candy and maker 1 charges $0.50 per piece.

In this case, the total cost of the order = 3 * 1.00 + 8 * 1.00 + 6 * .50 + 20 * 1.00 = $34

The above is only an example of a **pieces**, a **makers** and a **prices** array. Suppose you are given another set of **pieces**, **makers**, and **prices** arrays.

(a) (3 points) What are the types of **pieces**, **makers** and **prices**?

(b) (4 points) Find **C** and **M**. Using the given arrays, write the code to declare and initialize **C** and **M**.

(c) (23 points) Write code to find the total cost of the order. Your code should do the following:
Declare and initialize a variable that represents the total cost of the order.
For each kind of candy in **pieces** calculate the cost of the entry by

1. finding the maker of the candy in **makers**
2. finding the per piece price for the candy based on the maker in **prices**
3. multiplying the per piece price by the number pieces indicated in found in **pieces**
4. adding to the total cost

Print the total cost.