CSC 300 Spring 2019 - Lytinen
Final Exam
June, 2019

NAME:

Directions:

This is a take-home exam. It is worth 30% of your overall grade for the quarter, and will be graded on a scale of 0 to 30. There are 6 problems; note the point value of each. You must post your answers into the D2L dropbox by **Saturday, June 15 at 11:59 pm.**

The exam is **open book and open notes.** You may use any computing devices such as a laptop or a tablet, and you may not use your cell phone for any purposes at all. You must work alone, which means that you may not communicate with anyone (live or electronically) during the exam. Plagiarism of any kind is not allowed. Plagiarism is defined in the DePaul Academic Integrity policy as ``any use of words, ideas, or other work products attributed to an identifiable source, without attributing the work to the source from which it was obtained." For example, copying an answer from the Internet is plagiarism.

Problems

1. (6 points total) Explain which data structures that we have discussed during this quarter would be most useful or best describe the following applications. You may choose from the following data structures: stack, queue, list, set, or priority queue. Choose only one data structure per question.

    a. (2 points) You are building a system which maintains a class roster. Students may add or drop the class at any time before the quarter begins. The order of students in the roster is not important. Assume the roster never becomes full.

    b. (2 points) You are building a system which maintains a wait-list of students wanting to register for a closed course. If spaces in the course open up, students are added to the course's roster according to the number of credit hours they have.

    c. (2 points) You work at a grocery store. Your job is to restock the shelves. When you place new items onto the shelves, they go in front of any remaining items of the same type. Customers also take items that they want from the front of the shelves.

2. (6 points total) During our discussion of lists, I presented an array-based ``wrap-around'' version of a list. The final version was called **ArrayListWrap300**. The class had the following instance variables:

```
private T[] items;      // the data in the list is stored in this array
int front, back, size;  // front and back are the indices of the first item
                        // and the last item in the list, respectively
```

Here is a diagram depicting the state of an **ArrayListWrap300** object. Assume the **ArrayListWrap300** object is called **lst**.

```
-------------------------------------------------------------------
| "A"  | "T"  |  "A"  | null  | null | null  | null | "D" |  front = 7, back = 2, size = 4
-------------------------------------------------------------------
```

a. (2 points) What would print as a result of executing the statement `System.out.println(lst)`? Explain your answer.

b. (2 points) Draw a diagram which depicts the same list as a linked list.

c. (2 points) Draw a diagram (array-based) to depict the state of the instance variables of **lst** (including the underlying array) after execution of the statement lst.remove(1) (i.e., remove the first “A”). Your diagram should depict the items stored in the array, and the values of front, back, and size.

3. (4 points total) Give the Theta-complexity of the functions below.

   a. (2 points) $f(n) = 2n + (n \log(n+1))$

   b. (2 points) $f(n) = n * (\log(n-1))/2$

4. (4 points total) Give the Theta-complexity of the loop constructions below as a function of n.

   a. (2 points)

```
int sum = 0;
for (int i=0; i<n; i++)
    sum++;
for (int j=n; j>0; j /= 2)
    sum++;
```

   b. (2 points)

```
int sum = 0;
for (int i=n; i>0; i--)
      for (int j=i; j<n; j *= 2)
            sum++;
```

5. (5 points total) Show the state of a priority queue (heap) which is initially empty, after each call below. Assume that priority is determined by ordering in the alphabet, and that the priority queue object is called **pq**.

    a. (1 point) pq.offer("e")

    b. (1 point) pq.offer("d")

    c. (1 point) pq.offer("c")

    d. (1 point) pq.offer("a")

    e. (1 point) pq.poll()

6. (5 points total) Assume that a hash set named **set** is initially empty, the set is implemented using an array of length 13, the hash function is *h(x)* = x % T (where T is the length of the array), and that quadratic probing is used if there are collisions. Show the state of the array after each operation below:

    a. (1 point) set.add(5);

    b. (1 point) set.add(19);

    c. (1 point) set.add(31);

    d. (1 point) set.add(14);

    e. (1 point) set.add(44);