Name: _____ Andrew Id: _____

15-121 Fall 2018 Quiz 7

Up to 20 minutes. No calculators, no notes, no books, no computers. Show your work!
There are questions on *both sides* of this paper.

1. **Short Answer**

   (a) (2 points) What does it mean for a sort to be stable?

   (b) (2 points) Describe what specifically about the selection sort algorithm makes it stable. (In other words, why is selection sort stable?)

   (c) (4 points) Assume I am building a simple calculator that has the following behavior. When I see an integer, I push its value onto a stack. When I see an operator (`+, -, *, /`), I pop the stack once, and store the result in a variable called `op2`. I then pop the stack a second time, and store the result in a variable called `op1`; I then perform the arithmetic operation specified by that operator (`op1` operator `op2`) and push the result of that operation back onto the stack.

   For example, if I entered `7 2 -` , `peek()` would return `5` since `7` would be pushed onto the stack, then `2`, and then the `-` would cause `2` to be popped and stored in `op2`, `7` to be popped and stored in `op1`, and the result of `7-2`, which is `5`, would be pushed onto the stack.

   Tell me what value is returned by peek() after the following sequence of values is processed. Write your final answer in the box, but show you work in the empty space next to the box.

   ```
   16 4 / 5 * 5 -
   ```

   

2. **Free Response**

Consider a linked list similar to that of Homework 5. Assume the following things are true about the list implementation:

- The list has both a `head` and a `tail` pointer.
- The `ListNode` has a `next` pointer and a `data` pointer. The `data` pointer is a `DataType` that implements `Comparable<DataType>`. (This means that the `DataType` has a `compareTo` method.)
- The methods `add(DataType value)` and `remove(DataType value)` are already implemented properly.
- The following method is also included:

```
public void sortList() {
        MyList<DataType> newList = new MyList<DataType>();

        DataType tmp = this.findMin();
        while (tmp != null) {
            this.remove(tmp);
            newList.add(tmp);
            tmp = this.findMin();
        }

        this.head = newList.head;
        this.last = newList.last;
}
```

(a) (10 points) Write the private helper function `findMin()` that returns the minimum value in the linked list or returns `null` if no such value exists.

(b) (2 points) Assuming your answer to part (a) works, what is the big-oh efficiency of `sortList()`?