Name________________________

CSCI 151

Practice Midterm

November 2021

This practice midterm has many questions for you to use to study for the exam. Note, this exam is way longer than the exam Professor Geitz is going to give, we have done that intentionally so that you have many questions to practice with.

Good Luck! You got this!

Q1. **Write a Java program that opens file “foo.txt”, makes a list, without repetitions, of all of the words it contains, and then prints this list in alphabetical order.** You can take the “words” of the file to be anything separated by whitespace. For example, the string “How’s this and this and this?” has 4 unique words: “How’s”, “this”, ‘”and”, “this?”. You will note that this is just the way the next( ) method of the Scanner class works.

You can ignore any import statements your program will need; Eclipse will take care of those. Note that Java’s ArrayList class has a contains method, where L.contains(x) is true if item x is an element of list L. To sort list L in Java you can use Collections.sort(L). This doesn’t return anything; it modifies the order of the elements in L.

Q2. What are the contents of list after these operations?

```
ArrayList list = new ArrayList();
list.add(1);
list.add(2);
list.add(0, 3);
list.add(2, 4);
list.add(1, 5);

list.get(1);
list.remove(4);

list.add(1, 7);
```

Q3. In Lab2 we used the declaration `public class MyArrayList<E> extends AbstractList<E>` so MyArrayList<E> is a subclass of Java's abstract class AbstractList<E>. On the other hand, the Java documentation says that AbstractList<E> implements the List<E> interface.

Explain what the difference is between an abstract class and an interface.

Q4. **Suppose S is a stack that starts off empty and we do the following sequence of operations:**

```
S.push(1);
S.push(2);
S.pop();
S.push(3));
```

a. **What will S.top() return:**

Now we continue with this same stack and do

```
        S.push(4);
        S.push(5);
```

b. **What will S.pop() return now:**

c. **What will another S.pop() return:**

d. **What will another S.pop() return:**

Suppose Q is an empty queue and we do:

```
        Q.enqueue(1);
        Q.enqueue(2);
        Q.dequeue();
        Q.enqueue(3));
```

e. **What will Q.front( ) return:**

Now we do two enqueues:

```
        Q.enqueue(4);
        Q.enqueue(5);
```

f. **What will Q.dequeue() return:**

g. **What will another Q.dequeue() return:**

h. **What will one more Q.dequeue() return:**

__Q5.__ **In this problem you need to give code for two Queue methods.** To keep this simple we will make a Queue that only holds integer values. Here is the start of the class declaration:

```
public class Queue {
      Node front, back;
      int size;

      class Node {
            int data;
            Node next;

            Node (int item) {

                  data = item;
                  next = null;

            }
      } // this ends the Node class

                  public Queue ( ) {
                        size = 0;
                        front = null;
                        back = null;
                  }…

} // end of the Queue class
```

Give code for the Queue's method

void enqueue(int value): This adds a new node with the given value to the Queue. State the runtime of this method

int dequeue( ) throws NoSuchElementException: This should return the value being removed from the Queue. It should throw the exception if some doofus tries to dequeue an already empty queue. State the runtime of this method.

<u>Q6.</u> Suppose you are implementing stacks with an ArrayList, as you did in Lab3. Let's say that data is the name of the ArrayList. You could implement push() and pop() by having items enter and leave at index 0 -- so push(x) become data.add(0, x) and pop() becomes data.remove(0) or you could have items enter and leave at the end of the list – so push(x) becomes data.add(x) and pop() becomes data.remove(data.size()-1) .

Does it matter which version you implement? Is one more efficient or more reliable than the other? Give some explanation for your answer.

<u>Q7.</u> Suppose now you choose to implement your stack with a LinkedList. Write the following methods.
a. void clear()

b. int peek()

Q8

a. Describe the role of the constructor

b. What is the java syntax for inheritance? How about to incorporate interfaces in a class?

c. What is the difference between the public and private modifiers?

Q9. **Write the Big-O runtime of the following functions**.

Runtime: ______________

```
public void clear(){
        return new int[] ;
}
```

Runtime: ______________

```
public void Stevie(int n){
        for (int i = 0; i <= n; i++){
                System.out.println( This is my + i + th day eating at Stevie)
        }
}
```

Runtime: ______________

```
public int foo(int[] sortedArray, int key, int low, int high) {
        int index = Integer.MAX_VALUE;
        while (low <= high) {
                int mid = (low + high) / 2;
                if (sortedArray[mid] < key) {
                        low = mid + 1;
                } else if (sortedArray[mid] > key) {
                        high = mid - 1;
                } else if (sortedArray[mid] == key) {
                        index = mid; break;
                }
        }
        return index;
}
```

Q10. Describe what the following code does. Think about it on a case-by-case basis:

```
public int OWLS(int input, String output) {
        int newIn;
        boolean outputFlag = false;

        if (output != ""){
                newIn = input += 1;
                outputFlag = true;
        } else {
                newIn = input;
        }

        If ((newIn % 2) == 0 && outputFlag){
                System.out.println("Gosh, I really appreciate my owls");
                System.out.println("input int: " + input);
        } else if ((newIn % 2) == 0 && !outputFlag){
                System.out.println("I miss the Jonas Brothers"};
                System.out.println("input int: " + newIn);
        } else {
                System.out.println("Its cool to study computer science"};
        }
}
```