Name_____

CSCI 151
Practice Midterm Solutions
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.

```
public class ExamQ2 {

	public static void main(String[] args) {

		ArrayList<String> A = new ArrayList<String>();

		try {

			Scanner scan = new Scanner(new File("foo.txt"));

			while (scan.hasNext()) {

				String w = scan.next();

				if (!A.contains(w))

					A.add(w);

			}

			Collections.sort(A);

			for (String w: A)

				System.out.println(w);

		} catch (FileNotFoundException e) {

			System.out.println(e);

		}

	}

}
```

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);
```

[3,7,5,1,4]

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.

An abstract class is a class in which at least one method is *abstract*, which means it has a header but no body.

An interface is just a list of function headers. Everything in an interface is abstract; there is no actual code in 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: 3

Now we continue with this same stack and do

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

b. What will S.pop() return now: 5

c. What will another S.pop() return: 4

d. What will another S.pop() return: 3

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: 2

Now we do two enqueues:

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

f. What will Q.dequeue() return: 2

g. What will another Q.dequeue() return: 3

h. What will one more Q.dequeue() return: 4

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:

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

```
public void enqueue(int value) {
        Node p = new Node(value);
        if (size == 0) {
                front = p;
                back = p;
        } else {
                back.next = p;
                back = p;
        }
        size += 1;
        }
```

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.

```
public int dequeue() throws NoSuchElementException {
        if (size == 0)
                throw new NoSuchElementException( "Empty queue" );
        else {
                Node p = front;
                front = front.next; size -= 1;
                return p.data;
        }
}
```

Q6. 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.

Both versions are reliable and will run correctly, but the version where we add and remove from index [0] is much less efficient. Each time we add an element at index [0] of an Array List we have to shift all of the remaining entires up one index. Each time we remove the element at index [0] we have to shift all of the remaining elements down one index. If the stack contains n elements, push() and pop() in this implementation are both O(n) operations. When we add and remove from the high end of the ArrayList push() and pop90- are both O(1) operations,

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

```
public void clear(){
        head.next = tail;
}
```

b. int peek()

```
public int peek(){
        return head.data or tail.data;
}
```

Q8

a. Describe the role of the constructor

The constructor's role is to define the properties of an object.

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

Use 'extends' for inheritance, use 'implements' to incorporate an interface

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

Abstract Class: used to provide common method implementation to all the subclasses or to provide default implementation.

Interface: An interface is a reference type in Java. When a class implements an interface, you can think of the class as signing a contract, agreeing to perform the specific behaviors of the interface

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

```
public int [] clear(){                                          Runtime: O(1)
        return new int[] ;
}

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

public int foo(int[] sortedArray, int key, int low, int high) {
        int index = Integer.MAX_VALUE;
        while (low <= high) {                                   Runtime: O(logn)
                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"};
        }
}
```

This code takes in an integer and a string. If the string is not empty, we add one to our input (and store it in newIn) and set our flag to true, otherwise we set newIn to the input number. Then, if newIn is even and our string is empty, we print “Gosh I really appreciate my owls” and the unmodified input number. Otherwise, if our newIn is even and our string is not empty, we print the line about the Jonas Brothers and the newIn integer. Otherwise, we print “it’s cool to study computer science” which it really is.