COMP 241Review

Topics:

- Java Review
  - Understanding and writing Java code, understanding references and memory, class design and scope.
- Big O notation
  - Algorithms: What is the big o notation?
  - Looking at a piece of code-what is the big o running time?
  - Compare two pieces of code or two big O running times-which is slower?
- Abstract data types
  - Using them, designing for them, and programming them.
  - Big O analysis for operations with different ADTs.
  - *Implementation* vs *interface*
- Linked Lists
  - Designing linked lists, using linked lists, variables/implementation (node, head, tail, etc.)
  - Singly-linked lists
  - Doubly-linked lists
  - Circular linked lists
- Stacks
  - Using stacks, designing stacks
- Queues
  - Using queues, designing queues

1. Using a Stack data type, write a Java function to reverse a String.

```
public static String ReverseString(String str)
  {
    String reverseString = “”;
    // Declare a stack of type Character
    Stack<Character> stack = new Stack<Character>();

    // Traverse the String and push the character one by
    // one into the Stack
    for (int i = 0; i < str.length(); i++) {
      // push the character into the Stack
      stack.push(str.charAt(i));
    }

    // Now Pop the Characters from the stack until it
    // becomes empty

    int i = 0;
    while (!stack.isEmpty()) { // pop elements until stack becomes empty
      // get the character from the top of the stack
      reverseString = reverseString + stack.pop();
      i++;
    }
    // return string object
    return reverseString;
  }
```

2. Given an int queue (named queueOfInts) with the elements {5, 12, 2, 44, 51, 1}, what would the following code print out? (Here, the 5 element is the front of the queue.)

```
public static void main(String[] args)
{
            queueOfInts.dequeue();
            queueOfInts.dequeue();
            queueOfInts.enqueue(22);
            System.out.println(queueOfInts.dequeue());
            System.out.println(queueOfInts.dequeue());
            System.out.println(queueOfInts.dequeue());



}
```

```
Output: 2

     44

     51
```

If we were asked about what was inside the queue after this function, we would say [1, 22]

Reasoning: We remove the first two items in the queue with the dequeue calls and then add 22 to the **end** of the queue. Then we use dequeue to print the first three items in the queue.

Make sure that you can perform this sort of operation with stacks, queues, and linked lists (of each type).

```
}
```

3. What is the big O running time for the following Java code snippets? (Make sure to put this in terms of n.)

a.
```
public static void main(String[] args)
{
        recursiveMultiply(90, 80);
}       O(n)

public int recursiveMultiply(int a, int b)
{
        If(b <= 0)
        {
                return 1;
        }
        return a * recursiveMultiply(a, b-1);
}
```

b.
```
for(int i = 0; I < n^2; i++)
{
        System.out.println("Looping!");
}       O(n^2)
```

c.
```
public static void main(String[] args)
{
        Weird(0);
}       O(2^n)

public String weird(int n)
{
        If(n > 10)
        {
                return "t";
        }
        return weird(n+1) + n + weird(n+2);
}
```

4. Suppose there exists a RLinkedList and Node class for a singly-linked list:

```
public class Node<E>
{
   public E data;
   public Node next = null;
}
public class RLinkedList<E>
{
   public Node<E> head;
}
```

Using this RLinkedList class, create a circular linked list by filling in the following methods: add(E data, int position), delete(int position), and getSize(). You may add additional variables or functions to this class as needed.

```
//Note that I haven't implemented any error checking for this class!

public class RCircularList<E>
{

    RLinkedList<E> circleData;
    int size = 0;

    public void add(E data, int position)
    {
            if(position == 0)
            {
                    Node<E> temp = head;
                    head = new Node<E>(data, temp);
                    temp = head;
                    for(int i = 0; i<size; i++)
                    {
                            temp = temp.next;
                    }
                    temp.next = head;
            }
            else
            {
                    Node<E> temp = head;
                    Node<E> prevReference = null;
                    for(int i = 0; i<position; i++)
                    {

                            prevReference = temp;
                            temp = temp.next;
                    }
                    prevReference.next = new Node<E>(data, temp);
            }
            size++;

    }
```

```
public void delete(int position)
    {//Your code here
            if(size <= 1)
            {
                    head = null;
            {
            if(position == 0)
            {
                    Node<E> temp = head;
                    head = head.next;
                    for(int i = 0; i<size; i++)
                    {
                            temp = temp.next;
                    }
                    temp.next = head;
            }
            else
            {
                    Node<E> temp = head;
                    Node<E> prevReference = null;
                    for(int i = 0; i<position; i++)
                    {

                            prevReference = temp;
                            temp = temp.next;
                    }
                    prevReference.next = temp.next;
            }
            size--;

    }

    public void getSize()
    {
            return size;

    }

}
```

What is the big O running time of each of the methods you've created?

Add – O(n)
Delete – O(n)
getSize – O(c)

5. There exists a data structure named RDataBank that stores the data a user enters in a String array named DataBankArray. This array initially starts with 13 elements, but once a user enters more than 13 values into the array it has to be resized. Write a function to resize the DataBankArray to twice its current size once the capacity has been exceeded.

```
public class RDataBank
{
        String[] DataBankArray = new String[13];

        //Your function here.
        private void resize()
        {
          int newSize = 2*DataBankArray.length;

          String[] newArray = new String[newSize];

          int i = 0;
          while (i < DataBankArray.length)
          {
            newArray[i] = DataBankArray [i];
            i++;
          }

          DataBankArray = newArray;
        }




}
```

What is the big O running time of this function?

O(n)

6. The company you work for needs an implementation of a Stack that uses a LinkedList to store data. Using the RLinkedList class from Problem 4, design a Stack class with a push, pop, and peek function. This class must work for multiple data types and so should use the *generic* functionality (implemented with <E> in Java). What is the big O running time of each of these functions?

Push – O(c)

Pop – O(c)

Peek – O(c)

```
// Stack implementation in Java
class RStack<E> {

  // store elements of stack
  private RLinkedList arr;

  // Creating a stack
  RStack() {
    // initialize the array
    // initialize the stack variables
    arr = new RLinkedList();
  }

  // push elements to the top of stack
  public void push(E x) {
    // insert element on top of stack
    System.out.println("Inserting " + x);
    arr.head = new Node<E>(x, arr.head);
  }

  // pop elements from top of stack
  public E pop() {

    // if stack is empty
    // no element to pop
    if (arr.head == null) {
      System.out.println("STACK EMPTY");
      // terminates the program
      System.exit(1);
    }
    E output = (E) arr.head.data;

    arr.head = arr.head.next;
    return output;
  }

  // pop elements from top of stack
  public E peek() {
        // if stack is empty
        // no element to pop
        if (arr.head == null) {
            return null;
        }

        // pop element from top of stack
        return (E) arr.head.data;
    }
}
```