CS 151 Spring 2020
OWLS Midterm Exam
March 4 2020

Name:

T-number:

prev ->next = toDelete ->next;
delete toDelete;
// if only forgetting were
// this easy for me.
SNIFF
assert "It's going to be okay.";

This test is closed notes, closed book, closed computer (no running anything in Java). No aids are permitted. In total there are a bunch of questions. You have 60 minutes. We will not be scanning these tests to grade them: it doesn't matter if you write your T-number on the top of every page but make sure you write your answers in the space provided for them. Are you actually reading this? We did not provide a few extra blank pages at the end for extra workspace; if you use the extra pages, please note so on the question(s) you use them for. Please also sign the honor code below.

Good luck!

Honor Code:

1. **Short Answer** (5 pts)

   (a) Describe the role of the constructor
   Answer: 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?
   Answer: Use "extends" for inheritance. Use "implements" for incorporation of an interface

   (c) In your own words, describe both an abstract class and an interface.
   Answer:
   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.

   (d) Describe how bubble sort works in your own words.
   Answer: Bubble sort, is a sorting algorithm that repeatedly steps through the list, compares adjacent elements and swaps them if they are in the wrong order.

   (e) What is the difference between the public and private modifiers?
   Answer: Public means you can access it anywhere while private means you can only access it inside its own class.

   (f) My favorite data structure is ________________

2. **ArrayLists** (16 pts) What are the contents of list after these operations?

```
ArrayList<Integer> list = new ArrayList<Integer>();
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);
```

Answer: [3,7,5,1,4]

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

   (a) (5pts)

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

   Answer: O(1)

   (b) (5pts) *Note:* "This is my " and "the day eating..." should be in quotes

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

   Answer: O(n)

   (c) (3 pts)

   ```
   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;
   }
   ```

   Answer: O(log(n))

4. **Stacks**

   a. What are the contents of stack after these operations?

      Stack<Integer> stack = new Stack<Integer>();
      stack.push(10);
      stack.push(5);
      stack.push(7);
      stack.push(11);
      stack.pop();
      stack.peek();
      stack.pop();
      stack.push(3);
      stack.push(7);
      stack.peek();
      stack.pop();
      stack.peek();
      Answer: [3,5,10]

   b. Suppose you choose to implement your stack with a linkedlist. Write the following methods.

      1. void clear

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

      2. int peek()

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

Describe what the following code does (so many points) 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 an string. If the string is 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 ”its cool to study computer science” which it really is.

5. **Queues**

a. Write a method that dequeues from a queue (using a doubly linked list implementation (assume the constructor initiliazes next and prev for each node, and that we have a setNext() and setPrev() function).

```
public T dequeue() {
   if(size==0){
      return null;
   }
   else {
      int x = tail.data;
      Node prevTail = tail.previous;

      tail = prevTail;
      tail.next = null;
      size--;
      return x;
   }
}
```

b. Give the BigOh runtime of 2 queue operations (ie peek, enqueue, dequeue. . . .)

Enqueue and Dequeue are both O(1)

7. (30 pts) **Short answer**

   (a) Explain how you would redesign a stack to support a getMin() function which returns the current smallest element in the stack. Give the runtime of this function

   Add a new variable to your data structure. Something along the lines of MinVal. Make it whatever is first put in the stack, and update it appropriately as smaller input gets passed in. Whenever getMin() is called, return MinVal.

   (b) Write about how you could use two stacks/queues and a linked list to implement one of the sorting algorithms we've studied. (Hint: return the *sorted list*)

   Dont worry about this question.

(c) Consider using a stack as a way of visualizing a recursive call. Implement a recursive fibonacci method to get the nth fibonacci number and draw out the stack after calling a recersive fibonacci function to get Fib(4).

```
static int fib(int n)
{
if (n <= 1)
  return n;
return fib(n-1) + fib(n-2);
}
```

Stack would be something along the lines of: fib(0) = 1, fib(1) = 1, fib(2) = 2, fib(3) = 3, fib(4) = 5.

(d) Consider a new data structure: the "Group Stack", which takes in integers and groups them as more repeat integers get added. In words, describe how you would implement a "push" operation.
Example: if our square stack was 0 1 2 3 4 and we pushed 3, 4, and then 5, our group stack would be 0 1 2 (3,3) 4, then 0 1 2 (3,3) (4,4), then 0 1 2 (3,3) (4,4) (5,5). Popping 2, 3, 3 off the stack would yield 0 1 (4,4) (5,5)

Make 10 stacks, one each for 0-9. As input comes in, add it to the appropriate stack. You could also implement this with 10 queues, or any combination of queues and stacks.

Thats it, you're done :)