Practice Exam 1

CIS 2168 Data Structures

Answer the questions in the spaces provided. **Please note** that there are no intentional errors in the code provided except in questions asking you to correct said code. Your written code does not have to be 100% syntactically correct.

Name: ______________________________

Page	Points	Score
3	6	
4	10	
5	10	
6	2	
7	10	
8	2	
9	10	
10	15	
11	15	
13	20	
Total:	100	

Useful notes:

- You are allowed to clarify any answer you give.
- You are allowed to ask for clarification.
- Things are never as complicated as they appear, especially the math.
- Never leave a question blank, even if you don't know the answer. We can't give partial credit to blanks.
- Extra credit is available for exceptional answers (up to five additional points).

Don't Panic

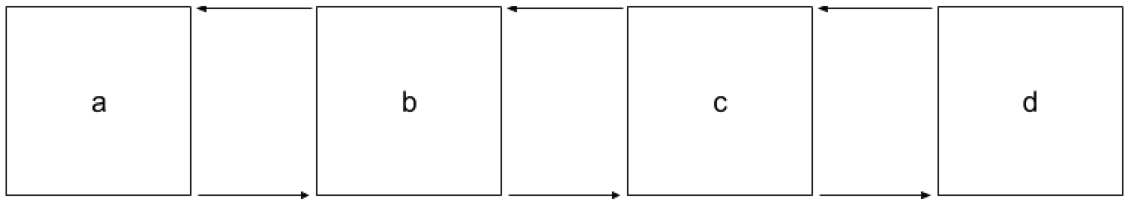

Figure 1: A doubly-linked list. Assume each node has a variable storing their memory location, denoted by the letter on the node.

1 Short Answer

For each of the following questions, please indicate which node is being referenced by the chain of variables.

1. (2 points) `d.prev.prev.next.prev`

   1. ______________

2. (2 points) `c.next.prev.prev`

   2. ______________

3. (2 points) `b.next.next.prev.prev.next.prev.next.prev`

   3. ______________

2 Exceptions

4. (10 points) Write a method that asks a user to input a `double`, retrieve that value using `Scanner.nextDouble()`, and print out twice the value of the input. Use a try/catch block to ensure that your method handles any exceptions caused by the user not entering a double.

Solution:

Put the `Scanner.nextDouble()` in a `try`. See your Assignment 1.

3 Lists

- x
- x
  - x
    - x
      - x
- x

5. (10 points) Suppose you have some `List` of Integers. Write a method that finds the minimum and maximum values stored in the list and returns their sums, regardless of implementation of the `List`.

```
public int minPlusMax(List<Integer> list){
        int min = Integer.MAX_VALUE;
        int max = Integer.MIN_VALUE;
```

```
        return min+max;
```

```
}
```

Solution:

Iterate through indices 0 to `list.size() -1`. `get()` the value at that index and update your mins and maxes accordingly.

6. (2 points) What is the time complexity of this algorithm?

6. ______________

7. (10 points) Suppose you have some `List` of Strings called `list` and a `String prefix`. Write a method that removes all the Strings from `list` that begin with `prefix`.

```
public void removePrefixStrings(List<String> list, String prefix){











}
```

Solution: This one is very very tough. Iterate through the list and use the `.remove()` method on any string `startsWith(prefix)`. To do it correctly, you must account for the fact that if you remove something in the middle of the list, the indices will shift!

8. (2 points) What is the time complexity of this algorithm?

8. ______________

______________ 2 points

4 Linked List

The following exercises deal with coding a `LinkedList`. Implement `add` and `remove`, found on the following pages.

- The `LinkedList` is composed of generic `Node` objects.
- The `LinkedList` contains a `Node<E> head` referencing the first node in the list.
- The `LinkedList` contains a `Node<E> tail` referencing the last node in the list.
- The `LinkedList` contains a method `size()` that returns the number of elements in the list.
- The `Node` objects are **doubly-linked**, and contain public variables `next` and `prev`, which reference the next and previous nodes in the list respectively.

9. (10 points) First, implement the helper function for `getNode`

```
// Returns the node at the specified index.
// Returns null if the index is out of bounds.
private Node<E> getNode(int index){
Node<E> found = null;




return found;
}
```

10. (15 points) Write the `add` method for a doubly-linked list. You may use the `getNode` method you just wrote. Be sure to handle all cases, such as an invalid index.

```
// adds the element to the list at the specified index
// returns true if adding is successful, false if adding fails
public void add(int index, E element) {
```

```
size++;
}
```

11. (15 points) Write the `remove` method for a doubly-linked list. You may use the `getNode` method you just wrote. Be sure to handle all cases, such as an invalid index.

```
// removes the element and node at index
// returns the item contained in the list at that index
public E remove(int index) {
E retval;    // to store the value to return
```

```
size--;
return retval;
}
```

Solution: Refer to code I did in class.

5 Analysis

12. (20 points) A `List` can be implemented in a number of ways. Compare and contrast an `ArrayList` and a `LinkedLists`. In which situations would you use one to implement a `List` over another?

Solution:

	ArrayList	Linked Lists
Add and Remove	In general, it takes $O(n)$ to add or remove something in an ArrayList. This is always the case if you add or remove something from the front or middle of a size n list, as you need to shift n or $\frac{n}{2}$ The exception to this is adding to the end of an ArrayList, which takes $O(1)$ time, except in the rare case that the ArrayList is full, in which case we need to reallocate, which takes $O(n)$ time.	If you are adding or removing to a location you have a reference to, such as the `head` or `tail` of an LinkedList, this takes $O(1)$ time. In all other cases, you need to search for a node at the location you are interested in, which takes $O(n)$ time.
Get and Set		
Memory Usage		

Refer to assignment 3.