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	8	
5	7	
6	12	
7	3	
8	12	
9	12	
10	10	
11	15	
12	20	
Total:	105	

Useful notes:

- You are allowed to clarify any answer you give.
  - All questions are essay questions, including the multiple choice.
- You are allowed to ask for clarification.
- Important `String` methods:
  - `length()`
  - `charAt(int index)`
  - `substring(int start, int end)`
  - `startsWith(String s)`
- `List` methods:
  - `get(int index)`
  - `set(int index, E item)`
  - `remove(int index)`
  - `add(E item)`
  - `add(int index, E item)`
  - `size()`
  - `contains(E item)`
  - `indexOf(E item)`
- 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

_____ 0 points

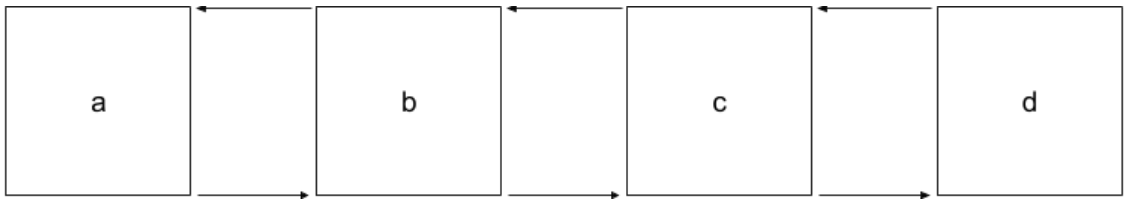

Figure 1: A doubly-linked list. Assume each node has a variable storing their memory location, denoted by the letter on the node.

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. _____

Cannot read field "prev" because "DoublyLinkedList.getNode(int).prev" is null

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

2. _____

Cannot read field "prev" because "DoublyLinkedList.getNode(int).prev" is null

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

3. a

 _____ 6 points

4. (2 points) Write a sequence of commands that will remove the tail of a doubly linked list.

```
if(index==size-1){
        tail = tail.prev;
        tail.next = null;
}
size--;
```

5. (2 points) Using a linked list with 3 nodes, draw what happens to the list as each line executes when you remove the tail.

6. (2 points) Write a sequence of commands that will add a new item to index 0 of a doubly linked list.

```
Node <E> adding = new Node<E>(item);
if(index==0){
        adding.next=head;
        head.prev=adding;
        head=adding;
}
size++;
```

7. (2 points) Using a linked list with 3 nodes, draw what happens to the list as each line executes.

_____ 8 points

2 Lists

8. (5 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;


    for(int i = 0; i<list.size();i++){
            if(min>=list.get(i)){
                    min=list.get(i);
            }
            if(max<=ist.get(i)){
                    max=list.get(i);
            }
    }
```

```
        return min+max;
}
```

9. (2 points) What is the time complexity of this algorithm?

O(n)

9. _____

_____ 7 points

10. (10 points) Suppose you have some `List` of `String`. Write a method `reverseWords` that returns a new `List` with all the Strings reversed.

```
// [hello, world, nice] -> [olleh, dlrow, ecin]
public List<String> reverseWords(List<String> list) {
```

```
// this is using the StringBuilder library
DoublyLinkedList <String> reverseWords = new DoublyLinkedList<>();
    for (int i = 0; i < words.size(); i++) {
       StringBuilder element = new StringBuilder(words.get(i));
       reverseWords.add(element.reverse().toString());
    }
    return reverseWords;
```

O(n)

```
// using double loop and going thought each element
for (int i = 0; i < words.size(); i++) {
        String element = "";
        for (int j = words.get(i).length()-1; j >= 0; j--) {
           element += words.get(i).charAt(j);
        }
        reverseWords.add(element);
     }
     return reverseWords;
```

O(n)

11. (2 points) What is the time complexity of this algorithm?

11. O(n)

3 Linked List

The following exercises deal with coding a `LinkedList`.

- 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.

12. (3 points) Write a method called `deleteList()`, which removes all the items in a list, making it empty.

```
// again, this is an instance method inside LinkedList
// so you have access to head, tail (optional), and the Node class
public void deleteList() {
```

```
while(head!=null){
        head = head.next;
        head.prev.next = null;
        head.prev = null;
}
```

```
while(size>0){
        this.remove(0);
}
```

 _____ 3 points

13. (10 points) Write an instance method called `count`. Count iterates over a `LinkedList` and returns the number of times `item` occurs in the list.

```
// This is an instance method, so you can access the head, tail, and the Node class
public int count(E item){

    int count = 0;
    Node<E> current = head;
    while(current!=null){
        if(current.item.equals(item)){
            count++;
        }
        current = current.next;
    }
```

14. (2 points) What is the time complexity of this algorithm? O(n)

14. _____

15. (10 points) Write a static method that takes in two **sorted** linked lists of integers and merges them into one sorted linked list and returns it. Since this a static method outside of `LinkedList`, you cannot access the Nodes. You are allowed to remove items from `listA` and `listB`

```
public static LinkedList<Integer> merge(List<Integer> listA, List<Integer> listB){
```

```
DoublyLinkedList <Integer> all = new DoublyLinkedList<>();
    while (listA.size()>0 && listB.size()>0){
      if(listA.get(0)<= listB.get(0)){
        all.add(listA.remove(0));

      }else{
        all.add(listB.remove(0));
      }
    }
    while (listA.size()!=0){
      all.add(listA.remove(0));
    }
    while (listB.size()!=0){
      all.add(listB.remove(0));
    }

    return all;
```

16. (2 points) What is the time complexity of this algorithm?

O(n)

16. _____

_____ 12 points

17. (10 points) Write a method that reverses a doubly-linked `LinkedList`. This method will be an instance method for `LinkedList`, so you can use the `Node head` and `tail` references and refer to the `LinkedList` object using the `this` keyword.

```
// You have access to the Node class
public void reverse(){
```

```
Node temp = null;
Node current = this.head;
while (current != null) {
   // swap the previous and next pointers of the current node
   temp = current.prev;
   current.prev = current.next;
   current.next = temp;

   // move to the next node
   current = current.prev;
}
// swap the head and tail pointers of the LinkedList
temp = this.head;
this.head = this.tail;
this.tail = temp;
```

 _____ 10 points

18. (15 points) Suppose we had a new type of `LinkedList`, called the `SortedLinkedList`, which is a linked list, but it keeps all the items in the list sorted. As a result, when we add an item to a `SortedLinkedList`, we don't provide an index, as the `SortedLinkedList` figures out where to put the new item based on the values already in the list.

    Your task is is to complete the `add` method for a `SortedLinkedList`, shown below. This method inserts a new item into the `SortedLinkedList` in such a way that the list remains sorted. For example, if the list is $[1, 2, 5]$ and we call `add(4)`, the becomes $[1, 2, 4, 5]$.

    For simplicity:

    - You can use either a singly or doubly linked list.
    - you can use `<`, `>`, and `==` to compare items, but if you remember how to use `compareTo()`, you can do so for extra credit.
    - You may not call the `add(int index , E item)` method.
    - You may not call the `getNode()` method (although you may rewrite it).
    - Your solution must run in `O(n)` time.

```
public void add(E item){
```

 _____ 15 points

4 Analysis

19. (20 points) A `List` can be implemented in a number of ways. Compare and contrast an `ArrayList` and a `LinkedList`. In which situations would you use one to implement a `List` over another?

```
d.remove(3)
```


_____ 20 points