CS367 Midterm Example Questions

1) Suppose `myList` is a `List` that initially contains `Strings` in this order: `["A","B","C","D","E","F"]`

Consider the following code fragment:

```
for (int i = 0; i < myList.size(); i++)
    myList.add(myList.remove(i));
```

After the code fragment executes, `myList` will contain `Strings` in which one of the following orders?

A. `["A","B","C","D","E","F"]`

B. `["B","D","F","A","C","E"]`

C. `["B","D","F","C","A","E"]`

D. `["B","D","F","C","E","A"]`

E. `["F","E","D","C","B","A"]`

2) What is printed when the following code is executed? Assume that `ArrayList` implements the List ADT interface.

```
List<Integer> myList = new ArrayList<Integer>();

for (int i = 7; i >= 0; i--)
    myList.add(i);

Iterator<Integer> iter = myList.iterator();

for (int i = 0; i < 10; i++) {
    try {
        System.out.print(iter.next());
    } catch (NoSuchElementException e) {
        System.out.print(" end list");
    }
}
```

A. `76543210`

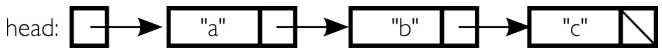
B. `01234567 end list`

C. `76543210 end list`

D. `01234567 end list end list`

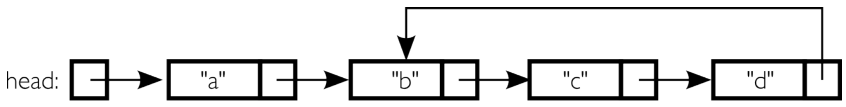
E. `76543210 end list end list`

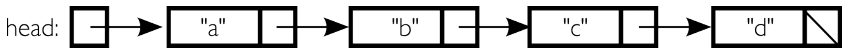
3) Given the singly linked list represented graphically below:

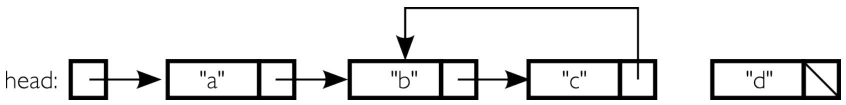


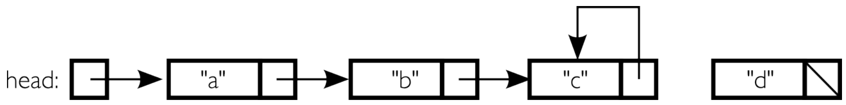
How will the list look after the following code is executed:

```
Listnode<String> temp = new Listnode<String>("d");
head.getNext().getNext().setNext(temp);
temp.setNext(head.getNext());
```

A. 

B. 

C. 

D. 

E. A null pointer exception

For the next two questions assume you are using the following data structure to implement a List: a singly linked list with a reference to the head of the linked list (no other references).

4) What is the worst-case complexity for add(item)? A. O(1) B. O(N)

5) What is the worst-case complexity for add(0,item)? A. O(1) B. O(N)

Written Question 1 a)

1) Assume that a `LinkedList` class has been implemented using a singly linked list with a dummy header node. The `LinkedList` class has the following fields:

```
private Listnode<E> items; // reference to the dummy header node
private int numItems;
```

Also assume that the items in a list can be compared with the `equals` method. Below is an implementation of the contains method for the `LinkedList` class:

```
public boolean contains(E item) {
    if (isEmpty()) return false;
    boolean found = false;
    int pos = 0;
    while (!found && pos < numItems) {
        found = get(pos).equals(item);
        pos++;
    }
    return found;
}
```

Part a) Give the worst-case complexity for the `contains` method above in terms of the problem size, N. **Briefly justify your answer**, including any assumptions you make about the complexity of any methods that are called by the `contains` method.

Written Question 1 b)

Part b) Write a second version of `contains` that functions the same as the `contains` method shown on the previous page and works directly with the links in the chain of nodes. To receive full credit, your `contains` method must not call any other List methods (such as `get()`).