1. Suppose you are given a singly-linked list class that holds strings and that maintains pointers to both the head and the tail of the list. Its fields and constructors are as follows:

```
public class SinglyLinkedList {
    protected ListNode head;
    protected ListNode tail;

    public SinglyLinkedList() {
            this.head = null;
            this.tail = null;
    }

    ...
}
```

   The ListNode class looks like this:

```
public class ListNode {
    private String value;
    private ListNode next;

    public ListNode(String value, ListNode next) {
            this.value = value;
            this.next = next;
    }

    public String getValue() {
            return this.value;
    }

    public ListNode getNext() {
            return this.next;
    }

    public String setValue(String newValue) {
            this.value = newValue;
    }

    public ListNode setNext(ListNode newNext) {
            this.next = newNext;
    }
}
```

Please add a new method to the class SinglyLinkedList with header:

```
public void keep(int howMany) {
```

which should modify the list so it only keeps the first `howMany` elements, dropping the rest of the elements from the list. E.g., if `myList` originally contains 10 elements, then executing `myList.keep(6)` should result in myList having only the first 6 elements of the list. You don't need to worry about keeping track of the discarded nodes as long as you cut them off from the rest of the list.

a. Write the pre- and post- conditions for the `keep` method. Just describe them in English.

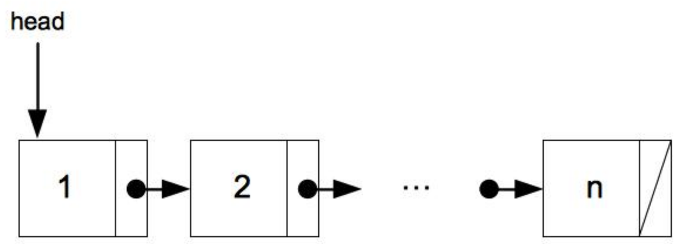
preconditions are things you assume are true before a method is called. postconditions are things that anyone that called this method can assume will have happened when the method finishes.

b. List at least one special case that either violates your preconditions or requires special handling.

c. Write the code for keep on the next page (you don't need to worry about comments). Remember that you should check your preconditions (you can use "RuntimeError" if you need to throw any exceptions).

```
public void keep(int howMany) {
```

2. You have a singly linked list with only a `head` pointer (see the figure below). The `insert()` method for the list inserts new values into the list so that the elements remain in sorted order using the obvious algorithm. In other words, after each insertion, the list is in sorted order.

   a. 

   Assume you are given a sequence of *n* values to insert one at a time into the list. What do you expect the total worst-case running time to be, using big-O notation, for inserting all of the values into the list? Give a brief (one to two sentence) **justification** for your answer.

   b. Suppose that the sequence of *n* values to be inserted just happen to be **in reverse sorted order**. E.g., you might be given the elements 47, 23, 19, 13, 7, 6, and finally 2. What do you expect the running time to be, using big-O notation, for inserting all of n values into the list? Give a brief (one to two sentence) **justification** for your answer.

5. Short answer

a. Describe carefully in words what happens when you insert an element into an ArrayList when it is already filled to capacity.

b. We noted that when using Java graphics, we must call the method repaint (which the programmer doesn't write) in order to get the computer to eventually call the method paint, which is the one the programmer actually writes. Please explain why this happens and what data structure that we have discussed in class is used to make this all work.

c. Explain how the run-time stack changes when a method is invoked and when the method completes execution.