**CIS163 Lab 11**

**Help for Final Exam**

**Preparation:**

Attended class and review your notes on Linked list.

**Objectives** (after completing the lab you will be able to do:)

* To write simple methods the modify a **Double** linked list
* Help you for the final exam.

**Activities:**

1. Cut and paste the code below into eclipse.
2. Complete the following methods. Note: It is a single linked list with a tail.

* public int getLen() {
* public void insertBefore (int index, String data) {
* public void insertAfter(int index, String data) {
* public String removeTop () {
* public boolean delAt(int index) {

1. Get checked off by the instructor
2. If you have time, write JUnits to test out your list.

public class Node {

private String data;

private Node next;

private Node prev;

public Node(String data, Node next) {

this.data = data;

this.next = next;

}

public Node() {

}

public String getData() {

return data;

}

public void setData(String data) {

this.data = data;

}

public void setNext(Node next) {

this.next = next;

}

public Node getNext() {

return next;

}

}

public class LinkListLab {

private Node top;

public LinkListLab() {

top = null;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Determines the size, that is, the number of elements in the list

\*

\* @return  the size of the list

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public int getLen() {

// place your code here

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Inserts a node before a specific index.  When the list is empty

\* that is, top = null, then the index must be 0. After the first

\* element is added, index must be:  0 <= index < size of list

\*

\* @param index a specific index into the list.

\*

\* @throws IllegalArgumentStringxception if index < 0 or

\* index >= size of the list

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public void insertBefore (int index, String data) {

// place your code here

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Inserts a node after a specific index.  When the list is empty

\* that is, top = null, then the index must be 0. After the first

\* element is added, index must be:  0 <= index < size of list

\*

\* @param index a specific index into the list.

\*

 \* @throws IllegalArgumentStringxception if index < 0 or

\* index >= size of the list

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public void insertAfter (int index, String data) {

// place your code here

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Removes the top element of the list

\*

\* @return returns the element that was removed.

\*

\* @throws RuntimeStringxception if top == null, that is,

\*  there is no list.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public String removeTop () {

return null;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* This Method removes a node at the specific index position.  The

\* first node is index 0.

\*

\* @param index the position in the linked list that is to be

\* removed.  The first position is zero.

\*

\* @throws IllegalArgumentStringxception if index < 0 or

\* index >= size of the list

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public boolean delAt(int index) {

return false;

}

// A simple testing program.  Not complete but a good start.

public static void main (String[] args){

LinkListLab list = new LinkListLab();

list.display();

System.out.println ("Current length = " + list.getLen());

list.insertBefore(0, "apple");

list.insertBefore(0, "pear");

list.insertBefore(1, "peach");

list.insertBefore(1, "cherry");

list.insertBefore(3, "donut");

list.insertAfter(0, "apple");

list.insertAfter(0, "pear");

list.insertAfter(1, "peach");

list.insertAfter(1, "cherry");

list.insertAfter(3, "donut");

list.display();

list.removeTop();

list.delAt(4);

list.delAt(2);

list.delAt(0);

list.removeTop();

list.removeTop();

list.display();

}

public void display() {

Node temp = top;

System.out.println ("\_\_\_\_\_\_\_\_\_\_\_ List \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

while (temp != null) {

System.out.println (temp.getData());

temp = temp.getNext();

}

}

}