CSC 212 Homework # 2 Solution of Selected Prob elms
ADT List & Double Linked List

Problem 1

1.

2. Write method `insertAll` as user of ADT List that takes two lists `l1`, `l2` and index `i` and insert all elements in `l2` in `l1` after position `i`. The list `l2` must not be changed. The first element has position 0, and assume that `i` is a valid position.

   Example 1.1. *If* $l1 : A, B, C, D,$ *and* $l2 : X, Z,$ *then after calling* `insertList( l1, l2, 1)`, *then* $l1 : A, B, X, Z, C, D.$

   Method signature **public** `<T>` **void** `insertAll(List<T> l1, List<T> l2,` **int** `i)`.

```
public <T> void insertAll(List<T> l1, List<T> l2, int i) {
        if (l2.empty()) {
                return;
        }
        l1.findFirst();
        for (int j = 0; j < i; j++) {
                l1.findNext();
        }
        l2.findFirst();
        while (!l2.last()) {
                l1.insert(l2.retrieve());
                l2.findNext();
        }
        l1.insert(l2.retrieve());
}
```

. . . . . .

Problem 2

1. Implement the following methods in the class `LinkedList`:

   (a) **Procedure** insertBeforeCurrent(T e). **Requires**: The list l should not be full. **Results**: The new element e is inserted before the current and the new element is made the current.

```
public void insertBeforeCurrent(T e) {
        Node<T> tmp = new Node<T>(e);
        if ((head == null) || (current == head)) {
                tmp.next = head;
                current = head = tmp;
        } else {
                Node<T> p = head;
                while (p.next != current) {
                        p = p.next;
                }
                tmp.next = current;
                p.next = tmp;
                current = tmp;
        }
}
```

2. Write the method `removeEvenElems`, member of the class `ArrayList`, that removes all the elements having an even position (the position of the first element is 0). The method must run in $O(n)$. The method signature is: **public void** `removeEvenElems ()`. **Do not call any methods and do not use any auxiliary data structure**.

   Example 2.1. *If* $l : A, B, C, D, E$, *then after calling the method* `l.removeEvenElems ()` l *becomes:* B, D.

```
public void removeEvenElems() {
        for(int i = 1; i < size; i += 2) {
                data[i / 2] = data[i];
        }
        size /= 2;
        if (size == 0) {
                current  = -1;
        } else {
                current = 0; //current is set to 0 in all cases
                   just to simplify the solution. It should be
                   set to 0 only if the element indexed at
                   current is removed.
        }
}
```

```
        public void removeEvenElem() {
                int indexIn = 0, indexShift = 1, cpt = 0;
                if (current % 2 == 0)
```

```
                        current = 0; // current will be removed,
                            so set current to 0
                while (indexShift < size) {
                        if (indexShift % 2 != 0) {
                                nodes[indexIn] = nodes[
                                   indexShift];
                                if (current == indexShift)
                                        current = indexIn;
                                cpt++;
                                indexIn++;
                        }
                        indexShift++;
                }
                size = size - cpt;
        }
```

```
public void removeEvenElems ()
{

if(size==1){
siz=0;
current=-1;
return;}

int j=1;
int n=size/2;
for (int i=0;i<n;i++)
{
nodes[i]=nodes[j];
j+=2;
}

if(current %2==0 && current==size-1)
      current=0;

 else
    current=current/2;

size=size/2 //or size-=(size+1)/2;

}
```

. . . . . .

Problem 3

1. Write the method `checkListEndsSymmetry` that receives a double linked list and an integer number k. The method checks if the double linked list has identical k elements going forward from the first element and backwards from the last one. The method returns *true* if they are identical, and *false* otherwise. The method

signature is: **public** <T> **boolean** checkListEndsSymmetry(DoubleLinkedList<T> dl, **int** k)

Example 3.1. *If* $dl = A \leftrightarrow B \leftrightarrow C \leftrightarrow D \leftrightarrow B \leftrightarrow A$ *and* $k = 2$*, then the method should return* true*. If* $k = 3$*, it should return* false*, since* C *does not equal* D*.*

```
public <T> boolean checkListEndsSymmetry(DoubleLinkedList<T> dl,
    int k) {
        LinkedList<T> l = new LinkedList<T>();
        dl.findFirst();
        for (int i = 0; i < k; i++) {
                l.insert(dl.retrieve());
                dl.findNext();
        }
        while (!dl.last()) {
                dl.findNext();
        }
        l.findFirst();
        for (int i = 0; i < k; i++) {
                T e1 = l.retrieve();
                T e2 = dl.retrieve();
                if (!e1.equals(e2)) {
                        return false;
                }
                l.findNext();
                dl.findPrevious();
        }
        return true;
}
```

.....