**ITRW222 18 September**

Test 8

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Student Nr:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Ontwerp ’n metode vir die GENERIC klas SLL (onder gedruk) genaamd: *deleteHead()* wat die eerste posisie van ’n geskakelde lys uitvee en die data waarde van daardie element terugstuur. Doen al 4 stappe!
2. Ontwerp ‘n metode vir die klas SLL genaamd: reverseList() wat ‘n SLL terugstuur wat in die omgekeerde volgorde is. LW die oorspronklike lys moet steeds bestaan.
3. Skryf ‘n toetsprogram wat hierdie metodes DEEGLIK toets met Integers.
4. Gebruik die 4 stap ontwerp metode (diagram; algemene gevalle; spesiale gevalle; en kode)

**Example /Voorbeeld**

Question 1:

InitialList: {2;4;1;6}

ans =InitialList. deleteHead() should be 2

Initial list should by (4;1;6}

Question2:

InitialList: (mylist) {2;4;1;6}

newList= InitialList.reverseList()

newList must be {6;1;4;2}

InitialList: {2;4;1;6}

1. Design a method for the generic class SLL (printed below) named: *deleteHead()*  that will remove the first position of the linked list and return the data-value of that position. Do all 4 steps.
2. Design a method for the class SLL named: reverseList() that returns a SLL in reverse order. Note that the original list must still exist.
3. Write a test program that thoroughly tests these methods with Integers.
4. Use the 4 step design method (diagram; general cases; special cases; and code)

public class SLL<T extends Comparable<? super T>>

{

private Element<T> head; private Element<T> tail;

public SLL()

{ head = null;

tail = null;

}

// available methods to use

public boolean prepend(T newElement) //insert at head

public boolean append(T newElement) //insert at tail

public String toString()

public class Element<T extends

Comparable<? super T>>

{

private T data;

private Element<T> next;

public Element(T param)

{

data = param;

}

}// end of inner class Node

}//end SLL outer class