**PRC3 Programming in C++**

**Assignment 8: “The recycle factory” (Linked Lists)**

In this exercise you will exercise manipulating linked lists.

**Recycling administration**

In a factory, bottles are recycled. The company's management wants to know how often each bottle returns to the factory. In preparation, all bottles have already been given a unique and never changing serial number.

On return to the factory, each bottle is scanned; the resulting serial numbers are maintained in class ScanList. The software designer decides to use a linked list to store all data. The definition for classes Scan and ScanList are given below.

It is up to you to implement Scan and ScanList using an ordered linked list (i.e.: the list is always sorted to the serial number).

Proof that all your methods work without errors using unit tests in the Google Test framework. Do test special cases such as: starting with an empty list; inserting new elements in the middle of the list (without breaking links), keeping the list ordered and memory leaks.

Definition Scan.h:

#ifndef SCAN\_H

#define SCAN\_H

class Scan

{

private:

int serialNumber;

int timesRecycled;

Scan\* next;

public:

Scan(int number);

// pre: -

// post: serialNumber == number and timesRecycled == 0

virtual ~Scan();

// pre: -

// post: object has been destructed

int getSerialNumber() const;

// pre: -

// post: serialNumber has been returned

Scan\* getNext();

// pre: -

// post: next scan has been returned

void setNext(Scan\* nextScan);

// pre: -

// post: next points to nextScan

void recycle();

// pre: -

// post: timesRecycled has been increased by one

int getTimesRecycled() const;

// pre: -

// post: timesRecycled has been returned

};

#endif

Definition ScanList.h:

#ifndef SCANLIST\_H

#define SCANLIST\_H

class ScanList

{

private:

Scan\* head; // pointer to linked list of Scans

public:

ScanList();

// pre: -

// post: list is empty

virtual ~ScanList();

// pre: -

// post: all scans have been destructed without memory leaks

void addScan (int serialNumber);

// pre: serialNumbers in the linked list are in ascending order

// post: IF serialNumber is not present in the list

// THEN a new scan with serialNumber has been added in the right place

// to the list

// ELSE timesRecycled has been increased by one

Scan\* getScanByNr(int position);

// pre: -

// post: IF 0 <= position < nr of scans in the list

// THEN return a pointer to the scan at that position

// ELSE return NULL

bool removeScan (int serialNumber);

// pre: serialNumbers are in ascending order in linked list scans

// post: IF serialNumber is present in the list

// THEN scan has been removed from the list and true has been returned

// ELSE false has been returned

int getTimesRecycled(int serialNumber);

// pre: serialNumbers in linked list scans are in ascending order

// post: IF a scan with serialNumber is present in the list

// THEN return the value of timesRecycled of that scan

// ELSE return 0

};

#endif