Labaration 5

drivers.h

```
#ifndef _DRVLST_
#define _DRVLST_

#include "queue.h"

//Checks input value
//Pre: True
//Post: Has returned the inputed integer
int mataIn();

//Adds en element to the queue
//Pre:  True
//Post: The element has been placed in the end of the queue
void laggTill(Queue<int> & Koe);

//Shows the entire queue on the screen
//Pre:  True
//Post: The entire queue has been printed to the screen
void visaKoe(Queue<int> & Koe);

//Shows the first element in the queue
//Pre:  True
//Post: The first element in the queue has been printed to the screen
void visaFirst(Queue<int> & Koe);

//Removes the first element in the queue
//Pre:  True
//Post: The first element in the queue has been removed
void tabortFirst(Queue<int> & Koe);

//Delets all the elements in the queue
//Pre:  True
//Post: Has deleted all the elements in the queue
void toem(Queue<int> & Koe);

//Show the status of the queue
//Pre:  True
//Post: Has printed the number of elemets in the queue
//      if the list is not empty, otherwise has printed
//      "Kön är tom".
void status(Queue<int> & Koe);


#endif
```

drivers.cpp

```
#include <iostream>
#include "drivers.h"
```

```
#include <iomanip>
using namespace std;


int mataIn()
{
  char buffer[100];
  int varde;
  bool felKoll = false;
  do
    {
      if(felKoll == true)
        cout << "Felaktigt värde! Försök igen: ";
      cin.getline(buffer, 100);
      if(buffer[0]=='0')
        return 0;
          felKoll = true;
    }while(!(varde=atoi(buffer)));
  return varde;
}


void laggTill(Queue<int> & Koe)
{

  int inmat;

  cout << "Lägg element i kö" << endl
       << "=================" << endl << endl;

  cout << "Ange det HELTAL du vill lägga till i kön: ";

  inmat = mataIn();

  Koe.enqueue(inmat);

  cout << "Elementet finns nu i kön!"<< endl << endl;

}

void tabortFirst(Queue<int> & Koe)
{

  int pos;

  cout << "Ta bort första elemetet i kön" << endl
       << "============================" << endl << endl;


  if(!Koe.isEmpty())
    {
      Koe.dequeue();

      cout << endl << "Första elementet i kön har raderats." << endl << endl;
    }
  else
    {
```

```
      cout << "Kön är tom! Går ej radera från tom kö." << endl << endl;
    }
}

void visaFirst(Queue<int> & Koe)
{

  cout << "Visa första elementet i kön" << endl
       << "==========================" << endl << endl;

  if(!Koe.isEmpty())
    {
      cout << endl << "Första elementet är " << Koe.first() << endl << endl;

    }
  else
    {
      cout << "Kön är tom! Går ej hämta värde." << endl << endl;
    }
}

void visaKoe(Queue<int> & Koe)
{

  int totKo[Koe.getSize()];

  cout << "Visa kö" << endl
       << "=======" << endl << endl;

  if(!Koe.isEmpty())
    {

      Koe.display(totKo);

      cout << "   Kö" << endl
           << " ----" << endl;

      for(int i = 0; i < Koe.getSize(); i++)
        cout << "   "  << totKo[i] << endl;

      cout << endl << endl;

    }
  else
    {
      cout << "Kön är tom! Går ej visa kön." << endl << endl;
    }

}


void toem(Queue<int> & Koe)
{

  int stor;

  cout << "Töm kön" << endl
```

```
       << "=======" << endl << endl;

  if(!Koe.isEmpty())
    {

      while(!Koe.isEmpty())
       Koe.dequeue();

      cout << endl << endl << "Kön har tömts" << endl << endl;

    }
  else
    {
      cout << "Kön är redan tom!" << endl << endl;
    }

}

void status(Queue<int> & Koe)
{
  cout << "Köns status" << endl
       << "==============" << endl << endl;

  if(Koe.isEmpty())
    cout << "Kön är tom!" << endl << endl;
  else
    cout << "Kön innehåller " << Koe.getSize() << " element." << endl << endl;

}
```

labb5.cpp

```
#include <iostream>
#include "drivers.h"
using namespace std;

//Skriver en meny på skärmen
//Pre:  True
//Post: Skrivit en meny på skärmen
void meny()
{

  cout << "  MENY  " << endl
       << "========" << endl
       << "1. Visa kö" << endl
       << "2. Lägg element i kö" << endl
       << "3. Visa första elementet i kön" << endl
       << "4. Ta bort första elementet i kön" << endl
       << "5. Visa köns status" << endl
       << "6. Töm kön" << endl
       << "0. Avsluta" << endl << endl
       << "Ange menyval (0-6): ";

}

int main()
```

```
{

  Queue<int> Ko;

  int menval;

  cout << "Köprogrammet   v. 1.0" << endl
       << "====================" << endl << endl
       << "Skrivet av:  Mathias Andersson"  << endl
       << "             Henrik Bäck" << endl << endl;


  while(menval != 0)
    {
  meny();

  menval = mataIn();

  cout << endl;

      switch(menval)
        {
        case 1: visaKoe(Ko);break;
        case 2: laggTill(Ko);break;
        case 3: visaFirst(Ko); break;
        case 4: tabortFirst(Ko); break;
        case 5: status(Ko); break;
        case 6: toem(Ko); break;
        case 0: break;
        default: cout << "Ogiltigt val!" << endl;
        }
    }


}
```

List.h

```
#ifndef _LISTPRG_
#define _LISTPRG_

#include "Node.cpp"

template <class T>
class List
{

 public:

  //Skapar en ett nytt obejt av klassen List
  //Pre: True
  //Post: Skapat ett nytt obejkt av klassen List
  List();

  //Tar bort dynamiskt allokerat minne
```

```
  //Pre:
  //Post: Tagit bort dynamiskt allokerat minne
  ~List();

  //Lägger till ett nytt element
  //Pre: 1 <= position <= size() + 1
  //Post: Elementet har lagts till i listan på position pos
  void insert(int pos, T varde);

  //Tar bort ett element
  //Pre: Elementet finns i listan
  //Post: Elementet har tagits bort
  void remove(int pos);

  //Sortera listan i stigande ordning
  //Pre: True
  //Post: Listan är sorterad i stigande ordning
  void sort();

  //Kontrollerar ifall listan är tom
  //Pre: True
  //Post: Returnerar true om listan är tom, annars false
  bool isEmpty();

  //Kontrollerar om ett element finns med i listan
  //Pre: True
  //Post: Returnerat true om elementet finsn i listan annars false
  bool isElement(T varde);

  //Hämta positionen för ett visst värde
  //Pre: True
  //Post: Returnerat det första värdet som upfyller 'varde'
  int getPos(T varde);

  //Returnerar antal element i listan
  //Pre: True
  //Post: Returnerat antalet element i listan
  int size();

  //Returnerar värdet på positionen pos
  //Pre: 1 <= pos <= size();
  //Post: Returnerat värdet på positionen pos
  T getElement(int pos);

 private:

  Node<T> * head;
  int sizen;

};

#include "List.cpp"

#endif
```

List.cpp

```
template <class T>
List<T>::List()
{

  head = 0;
  sizen = 0;

}

template <class T>
List<T>::~List()
{

  int stor;

  stor = sizen;

  for(int i = 1; i <= stor; i++)
    remove(1);

}

template <class T>
void List<T>::insert(int pos, T varde)
{
  Node<T> * newNode = new Node<T>();
  newNode -> data = varde;

  if(pos == 1)
    {
      newNode -> next = head;
      head = newNode;
    }
  else
    {
      Node<T> * temp = head;

      for(int i = 1; i < pos - 1; i++)
        {
          temp = temp -> next;
        }

      newNode -> next = temp -> next;
      temp -> next = newNode;

    }

  sizen ++;

}

template <class T>
void List<T>::remove(int pos)
{
```

```
  Node<T> * temp;
  Node<T> * temp2;

  temp = head;

  if(pos == 1)
    {
      temp2 = temp -> next;
      delete temp;
      head = temp2;
    }
  else
    {
      for(int i = 1; i < pos - 1; i++)
        temp = temp -> next;

      temp2 = temp -> next;
      temp2 = temp2 -> next;

      delete temp -> next;

      temp -> next = temp2;

    }
  sizen--;

}

template <class T>
void List<T>::sort()
{

  Node<T> * temp = head;
  Node<T> * temp2;
  Node<T> * temp3;



  for(int i = 1; i < sizen; i++)
    for(int k = 1; k < sizen; k++)
      {

        if(getElement(k) > getElement(k+1))
          {
            temp = head;
            if(k == 1)
              {
                temp2 = temp -> next;
                temp -> next = temp2 -> next;
                temp2 -> next = temp;
                head = temp2;
              }
            else
              {

                for(int j = 1; j < k-1; j++)
```

```
                {
                  temp = temp -> next;
                }

              temp2 = temp  -> next;
              temp3 = temp2 -> next;
              temp2 -> next = temp3 -> next;
              temp3 -> next = temp2;
              temp  -> next = temp3;

            }
        }
    }


}

template <class T>
bool List<T>::isEmpty()
{

  return !sizen;

}

template <class T>
bool List<T>::isElement(T varde)
{

  bool finns = false;

  for(int i = 1; i <= sizen; i++)
    {
      if(getElement(i) == varde)
        {
          finns = true;
          i = sizen + 1;
        }
    }

  return finns;

}

template <class T>
int List<T>::getPos(T varde)
{
  int position = 0;

  for(int i = 1; i <= sizen; i++)
    {
      if(getElement(i) == varde)
        {
          position = i;
          i = sizen + 1;
        }
    }
```

```
  return position;
}

template <class T>
int List<T>::size()
{

  return sizen;

}

template <class T>
T List<T>::getElement(int pos)
{

  Node<T> * temp;
  temp = head;

  for(int i = 1; i < pos ; i++)
    temp = temp -> next;

  return temp -> data;


}
```

Node.cpp

```
template <typename T>
class List;


template <typename T>
class Node
{
    friend class List<T>;
    private:
      T data;
      Node<T>* next;
};
```

queue.h

```
#ifndef __QUEUE_H__
#define __QUEUE_H__

#include "List.h"

//Q, Queue
//E, Element
//I, Integer
//B, Bool

template <class T>
class Queue
{
```

```
 public:


  //Ø → Q
  //Pre:  True
  //Post: Has created a new queue
  Queue();

  //Q → Ø
  //Pre:  !isEmpty()
  //Post: The queue has been destroyed
  ~Queue();

  //Q x E → Q
  //Pre:  True
  //Post: The element, data, has been enqueued
  void enqueue(T data);

  // Q → Q
  //Pre:  !isEmpty()
  //Post: The element has been removed
  void dequeue();

  //Q → E
  //Pre:  !isEmpty
  //Post: Has returned the first element in the queue
  T first();

  //Q → B
  //Pre:  True
  //Post: Has returned true if the list is empty otherwise false.
  bool isEmpty();

  //Q → I
  //Pre:  True
  //Post: Has returned the number of elemements in queue
  int getSize();

  //Q → Q
  //Pre:  !isEmpty
  //Post: The entire queue has been placed in the array allData
  void display(T * allData);

 private:
  List<T> queueList;
};

#include "queue.cpp"
#endif
```

queue.cpp

```
template <class T>
Queue<T>::Queue()
{
}
```

```
template <class T>
Queue<T>::~Queue()
{
}

template <class T>
void Queue<T>::enqueue(T data)
{

  queueList.insert(queueList.size()+1, data);

}


template <class T>
void Queue<T>::dequeue()
{

  queueList.remove(1);

}


template <class T>
T Queue<T>::first()
{

  return queueList.getElement(1);

}


template <class T>
bool Queue<T>::isEmpty()
{

  return queueList.isEmpty();

}


template <class T>
int Queue<T>::getSize()
{

  return queueList.size();

}


template <class T>
void Queue<T>::display(T * allData)
{

  for(int i = 1; i<=queueList.size(); i++)
    {
```

```
      *allData = queueList.getElement(i);
      allData++;
    }
}
```