Labaration 4

drivers.h

```
#ifndef _DRVLST_
#define _DRVLST_

#include "List.h"

//Ser till att inmatat värde endast kan anta siffra
//Pre: True
//Post: Returnerat inmatad siffra
int mataIn();

//Lägger till ett element i listan
//Pre:  True
//Post: Ett element tillagt i listan
void laggTill(List<int> & listan);

//Tar bort ett element från listan
//Pre:  True
//Post: Elementet på angiven position har raderats
void taBort(List<int> & listan);

//Sorterar listan stigande
//Pre:  True
//Post: Listan har blivit sorterad i stigande ordning
void sortera(List<int> & listan);

//Hämtar värdet på inmatad position
//Pre:  True
//Post: Ett värde har hämtats och skrivits till skrämen
void hamtaVarde(List<int> & listan);

//Skriver ut hela listan
//Pre:  True
//Post: Skrivit ut hela listan till skrämen
void visaLista(List<int> & listan);

//Söker efter ett element i listan
//Pre:  True
//Post: Skrivit ut elemetets position i listan på skärmen
void soek(List<int> & listan);

//Tömmer listan
//Pre:  True
//Post: Listan tömd
void toem(List<int> & listan);

//Kontrollerar om listan är tom
//Pre:  True
//Post: Skrivit ut listans status
void tom(List<int> & listan);
```

```
//Skrier ut antal positioner i listan
//Pre:  True
//Post: Skrivit ut antal positioner i listan.
void antal(List<int> & listan);

#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 << "Felaktikt värde! Försök igen: ";
      cin.getline(buffer, 100);
      if(buffer[0]=='0')
        return 0;
          felKoll = true;
    }while(!(varde=atoi(buffer)));
  return varde;
}


void laggTill(List<int> & listan)
{

  int inmat, pos;

  cout << "Lägg till element i listan" << endl
       << "==========================" << endl << endl;

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

  inmat = mataIn();

  cout << "Ange på vilken position du vill lägga värdet (1 till "
       << listan.size() + 1 << "): ";

  pos = mataIn();

  while((pos > listan.size() + 1 ) || (pos < 1))
    {
      cout << "Ogiltig position, ange ny: ";
      pos = mataIn();
    }
```

```
   listan.insert(pos, inmat);

   cout << "Det nya elemetet har lagrats!"<< endl << endl;


}

void taBort(List<int> & listan)
{

  int pos;

  cout << "Ta bort ett element från listan" << endl
       << "==============================" << endl << endl;


  if(!listan.isEmpty())
    {
      cout << "Ange position (1 till " << listan.size() << "): ";

      pos = mataIn();

      while((pos > listan.size()) || (pos < 1))
        {
          cout << "Ogiltig position, ange ny: ";
          pos = mataIn();
        }

      listan.remove(pos);

      cout << endl << "Elemetet på position " << pos << " har raderats" << endl <<
endl;
    }
  else
    {
      cout << "Listan är tom! Går ej radera från tom lista." << endl << endl;
    }
}

void sortera(List<int> & listan)
{

  cout << "Sortera listan" << endl
       << "==============" << endl << endl;

  if(!listan.isEmpty())
    {
      listan.sort();

      cout  << "Listan är sorterad i stigande ordning!" << endl << endl;
    }
  else
    {
      cout << "Listan är tom! Går ej sortera tom lista." << endl << endl;
    }

}
```

```
void hamtaVarde(List<int> & listan)
{

  int kokkos;

  cout << "Hämta element" << endl
       << "==============" << endl << endl;

  if(!listan.isEmpty())
    {
      cout  << "Ange position (1 till " << listan.size() << "): ";

      kokkos = mataIn();

      while((kokkos > listan.size()) || (kokkos < 1))
        {
          cout << "Ogiltig position, ange ny: ";
          kokkos = mataIn();
        }

      cout << endl << "Värdet på position " << kokkos << " är: " <<
listan.getElement(kokkos) << endl << endl;

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

void visaLista(List<int> & listan)
{

  cout << "Visa listan" << endl
       << "===========" << endl << endl;

  if(!listan.isEmpty())
    {

      cout << "Position\tVärde" << endl << endl;

      for(int i = 1; i <= listan.size(); i++)
        cout << setw(8) << i << "\t" << listan.getElement(i) << endl;

      cout << endl << endl;

    }
  else
    {
      cout << "Listan är tom! Går ej visa lista." << endl << endl;
    }

}

void soek(List<int> & listan)
{
```

```
  int sok;

  cout << "Sök ett element" << endl
       << "===============" << endl << endl;

  if(!listan.isEmpty())
    {
      cout << "Ange värde att söka efter: ";

      sok = mataIn();

      cout << endl;

      if(listan.isElement(sok))
        cout << sok << " finns på position " << listan.getPos(sok) << endl << endl;
      else
        cout << sok << " finns inte i listan!" << endl << endl;


    }
      else
        {
          cout << "Listan är tom! Går ej att söka." << endl << endl;
        }
}

void toem(List<int> & listan)
{

  int stor;

  cout << "Töm listan" << endl
       << "==========" << endl << endl;

  if(!listan.isEmpty())
    {
      stor = listan.size();

      for(int i = 1; i <= stor; i++)
        {
          listan.remove(1);
        }
      cout << "Listan är tömd!" << endl << endl;

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

}

void tom(List<int> & listan)
{
  cout << "Listans status" << endl
       << "==============" << endl << endl;
```

```
  if(listan.isEmpty())
    cout << "Listan är tom!" << endl << endl;
  else
    cout << "Listan innehåller element" << endl << endl;

}

void antal(List<int> & listan)
{

  cout << "Antal element i listan" << endl
       << "======================" << endl << endl;

  if(listan.isEmpty())
    cout << "Listan är tom, inga element finns!" << endl << endl;
  else
    cout << "Listan innehåller " << listan.size() << " st element" << endl <<
endl;


}
```

lab4.cpp

```
#include <iostream>
#include "List.h"
#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 listan" << endl
       << "2. Hämta element" << endl
       << "3. Lägg till element" << endl
       << "4. Ta bort element" << endl
       << "5. Sortera listan" << endl
       << "6. Sök efter element" << endl
       << "7. Töm listan" << endl
       << "8. Kontrollera listans status" << endl
       << "9. Antal element i listan" << endl
       << "0. Avsluta" << endl << endl
       << "Ange menyval (0-9): ";

}

int main()
{

  List<int> prgList;
```

```
  int menval;

  cout << "Listprogrammet  v.2.0" << endl
       << "====================" << endl << endl
       << "Skrivet av:  Mathias Andersson"  << endl
       << "             Henrik Bäck" << endl << endl;


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

  menval = mataIn();

  cout << endl;

      switch(menval)
        {
        case 1: visaLista(prgList);break;
        case 2: hamtaVarde(prgList);break;
        case 3: laggTill(prgList); break;
        case 4: taBort(prgList); break;
        case 5: sortera(prgList); break;
        case 6: soek(prgList);break;
        case 7: toem(prgList);break;
        case 8: tom(prgList);break;
        case 9: antal(prgList);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;


}
```