Bilaga A - Programkod

affar.cpp

```
#include "affar.h"

Affare::Affare()
{
  openChash();
  openChash();
}

Affare::~Affare()
{
}

void Affare::addToShop(char _namn[], int _alder)
{

  Person ny(_namn, _alder);
  shop.insert(1, ny);

}

int Affare::numChash()
{

  return kor.size();

}

void Affare::openChash()
{

  Queue<Person> ny;
  kor.insert(kor.size()+1,ny);

}

int Affare::numShopper()
{

  return shop.size();

}

std::string Affare::getShopperName(int number)
{

  std::string ny;
  ny = shop.getElement(number).getName();

  return ny;
}
```

```
int Affare::getShopperAge(int number)
{

  return  shop.getElement(number).getAge();

}


void Affare::viewQueue(Person * allPq, int queueNr)
{

  kor.getElement(queueNr).display(allPq);

}

int Affare::getQueueSize(int queueNr)
{

  return kor.getElement(queueNr).getSize();

}

void Affare::moveToQueue(int kundNr, int queueNr)
{

  std::string nameOld = getShopperName(kundNr);
  int alder = 0;
  char namn[30];

  strcpy(namn,nameOld.c_str());
  alder = getShopperAge(kundNr);
  Person ny(namn, alder);

  kor.getElement(queueNr).enqueue(ny);
  shop.remove(kundNr);

}

void Affare::closeChash(int queueNr)
{

  int antal = getQueueSize(queueNr);

  for(int i = 1; i <= antal; i++)
    {

      int kassa = getQueueSize(1);
      int t = 1, k = 0;

      for(k = 1; k <= numChash(); k++)
        {

          if((getQueueSize(k) <= kassa) && (k != queueNr))
            {
              kassa = getQueueSize(k);
              t=k;
```

```
            }

          }
        moveQueueToQueue(queueNr, t);

      }
  kor.remove(queueNr);

}

void Affare::moveQueueToQueue(int oldQueueNr, int queueNr)
{
  Person ny;
  ny = kor.getElement(oldQueueNr).first();
  kor.getElement(queueNr).enqueue(ny);

  kor.getElement(oldQueueNr).dequeue();
}

void Affare::servePerson(int queueNr)
{

  kor.getElement(queueNr).dequeue();

}
```

affar.h

```
#ifndef _affaren_12344_
#define _affaren_12344_

#include "queue.h"
#include "Person.h"
#include <cstring>

class Affare
{

 public:
  //Skapar ett nytt objekt affär
  //Pre:  True
  //Post: Nytt obejekt av klassen Affare har skapats
  Affare();

  //Destruktor
  //Pre:  True
  //Post: Objektet och all dess data har raderats från minnet
  ~Affare();

  //Lägger till en kund i affären
  //Pre:  _namn är max 30 tecken, 0 <= _alder =< 120
  //Post: En person har lagts till i butiken
  void addToShop(char _namn[], int _alder);


  //Flyttar en person från butiken till kön
```

```
//Pre:  Kunden med kundNr finns och Kön med queueNr finns.
//Post: En person har flyttats från butiken och ställts i kön.
void moveToQueue(int kundNr, int queueNr);

//Flyttar första personen i kön oldQueueNr till kön queueNr
//Pre:  Kön med nummer oldQueueNr finns och kön med nummer queueNr finns
//Post: Har flyttat första personen i kön oldQueueNr till kön queueNr
void moveQueueToQueue(int oldQueueNr, int queueNr);

//Betjänar första personen I kön queueNr och tar bort denne från kön.
//Pre:  Kö queueNr finns och innehåller minst 1 st person
//Post: Första kunden har betjänast och raderats från kön
void servePerson(int queueNr);

//Öppnar ny kassa-kö
//Pre:  True
//Post: En ny kassa-kö har öppnats
void openChash();

//Stänger en redan öppen kassa-kö
//Pre:  Kön med queueNr finns samt numChach() => 2
//Post: En kö har stängts och alla personer i kö har placerats i andra köer.
void closeChash(int queueNr);

//Returnerar antalet öppna kassor
//Pre:  True
//Post: Returnerat antalet öppna kassor
int numChash();

//Fyller en array med samtliga personer I butiken (ej i kö)
//Pre:  numShopper => 1, allCust är en array med numShopper platser
//Post: Den array som allCust pekar på har fyllts med personer
void viewShoppers(Person * allCust);

//Returnerar antalet personer i butiken (ej i kö)
//Pre:  True
//Post: Returnerat antalet personer i butiken, dock ej de i kö.
int numShopper();

//Returnerar namnet på en kund i affären
//Pre:  Kunden med kundnummer number finns i listan över kunder i butik
//Post: Namnet på kunden med kundnummer number har returnerats
std::string getShopperName(int number);

//Returnerar åldern på en kund i affären
//Pre:  Kunden med kundnummer number finns i listan över kunder i butik
//Post: Åldern på kunden med kundnummer number har returnerats
int getShopperAge(int number);

//Fyller allPq arrayn med personer från kö nummr queueNr
//Pre: allPq har getQueueSize(queueNr) antal platser och kö nummer queueNr finns
//Post: allPq innehåller personer från kö nummer queueNr
void viewQueue(Person * allPq, int queueNr);



//Returnerar storleken på en kö
```

```
  //Pre:  Kön med nummer queueNr finns
  //Post: Returnerat storleken på kön queueNr
  int getQueueSize(int queueNr);


 private:
  List<Person> shop;
  List<Queue<Person> > kor;

};

#endif
```

drivers.cpp

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

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


void kommerIn(Affare & shoppen)
{

  using namespace std;

  cout << "Någon kliver in i butiken" << endl
       <<"--------------------------" << endl << endl;

  char namn[NAMNLANGD];
  int alder = 0;
  char buffert[200];

  do
    {
      cout << "Vad heter personen? (30 tecken)" <<endl;
      cin.getline(buffert, 200);

      for(int i = 0; i < NAMNLANGD - 1 ;i++)
        namn[i] = buffert[i];
```

```
    }while((strlen(namn) == 0) || (namn[0] == '0'));


  do
    {
      cout << endl;

      if(alder<0 || alder>120)
        cout << "Felaktig ålder, försök igen!" << endl;

      cout << "Hur gammal är personen? (0-120 år)" << endl;
      alder = mataInInt();
    }while(alder<0 || alder >120);

  cout << endl << endl;

  shoppen.addToShop(namn, alder);


}

void gaTillKo(Affare & shoppen)
{

  using namespace std;

  cout << "Någon går till kassan" << endl
       << "---------------------" << endl << endl;

  int kundNr = 0;
  int koeNr = 0;
  string namn;

  if(shoppen.numShopper() != 0)
    {

      do
        {
          cout << "Ange kundnummer för personen som skall gå till kön (1 - "
               << shoppen.numShopper() << "): ";
          kundNr = mataInInt();
        }while(kundNr < 1 || kundNr > shoppen.numShopper());

      do
        {
          cout << "Ange kassa som personen skall gå till: (1 - "
               << shoppen.numChash() << "): ";
          koeNr = mataInInt();
        }while(koeNr < 1 || koeNr > shoppen.numChash());

      namn = shoppen.getShopperName(kundNr);

      shoppen.moveToQueue(kundNr, koeNr);

      cout << endl << namn  << " har gått till kassa-kö nummer "
           << koeNr << endl << endl;
```

```
    }
  else
    {
   cout << endl <<"Det finns ingen som kan gå till kassan, butiken är tom på
kunder. " <<  endl << endl;
    }

}

void visaKunder(Affare & shoppen)
{

  using namespace std;

   cout << "Minglar med kunderna" << endl
        << "--------------------" << endl << endl
        << setiosflags(ios::left) << setw(12) << "Kundnummer"
        << setw(NAMNLANGD+1) << "Namn" << setw(6)
        <<  "Ålder" << endl;

   for(int i = 1; i <= shoppen.numShopper(); i++)
     cout << setiosflags(ios::left) << setw(12) << i << setw(NAMNLANGD+1)
          << shoppen.getShopperName(i) << setw(6)
          << shoppen.getShopperAge(i) << endl;

   cout << endl << endl;


}

void oppnaKassa(Affare & shoppen)
{
  std::cout << "En ny kassa öppnar" << std::endl
            << "------------------"  << std::endl << std::endl
            << "PLING PLONG - Kassör till kassa "
            << shoppen.numChash()+1 << ", tack" << std::endl;

  Queue<Person> ny;

  shoppen.openChash();


  std::cout << "Kassa nummer " << shoppen.numChash()
            << " har öppnats. " << std::endl << std::endl;

}

void tittaKo(Affare & shoppen)
{

  using namespace std;

  int koeNr = 0;


  cout << "Titta på kassorna" << endl
```

```
       << "------------------" << endl << endl;

  do
    {
      cout << "Ange kassa du vill titta in i: (1 - "
           << shoppen.numChash() << "): ";
      koeNr = mataInInt();
    }while(koeNr < 1 || koeNr > shoppen.numChash());

  cout << endl << endl;

  Person tjock[shoppen.getQueueSize(koeNr)];

  cout << endl << "Tittar på kö nummer " << koeNr << endl << endl << endl
       << setiosflags(ios::left) << setw(13) << "Köplacering"
       << setw(NAMNLANGD+1) << "Namn" << setw(6)
       <<  "Ålder" << endl;

  shoppen.viewQueue(tjock, koeNr);

  for(int i = 0; i < shoppen.getQueueSize(koeNr); i++)
    cout << setiosflags(ios::left) << setw(13) << i+1 << setw(NAMNLANGD+1)
         << tjock[i].getName() << setw(6)
         << tjock[i].getAge() << endl;

  cout << endl << endl;


}

void stangKassa(Affare & shoppen)
{

  using namespace std;

  int kassa = 0;

  cout << "Stäng kassa" << endl
       << "-----------" << endl << endl;

  if(shoppen.numChash() != 1)
    {

      do{
      cout << "Vilken kassa skall st□ängas? (1 - " << shoppen.numChash()
           << ")"  << endl;
        kassa = mataInInt();
      }while(kassa < 1 || kassa > shoppen.numChash());

      shoppen.closeChash(kassa);

      cout << endl << "Kassan har stängts och alla kunder står nu i de andra
kassorna." << endl;

    }
  else
    {
```

```
      cout << "Det finns bara en kassa öppen, den måste få vara öppen för
kundernas skull.";
    }
      cout << endl << endl;


}

void betjana(Affare & shoppen)
{
  using namespace std;

  int kassa = 0;

  cout << "Betjäna någon ur kön" << endl
       << "--------------------" << endl << endl;

  cout << "Ur vilken kassa skall någon betjänas? (1 - "
       << shoppen.numChash() << ")"  << endl;

  do{
    kassa = mataInInt();
  }while(kassa < 1 || kassa > shoppen.numChash());


  if(shoppen.getQueueSize(kassa) > 0)
    {
      Person namnHamt[shoppen.getQueueSize(kassa)];
      shoppen.viewQueue(namnHamt, kassa);

      shoppen.servePerson(kassa);

      cout << endl << namnHamt[0].getName()
           << " har blvit betjänad och har gått hem jättenöjd!" << endl;

    }
  else
    {
      cout << "Kön är tom, ingen finns att betjäna.";
    }

cout << endl << endl;

}
```

drivers.h

```
#ifndef _AFFDRIVBAVER1_
#define _AFFDRIVBAVER1_

#include "affar.h"

const int NAMNLANGD = 30;

//Tar emot ett heltal
//Pre:  True
```

```
//Post: Returnerat ett heltal
int mataInInt();

//Lägger till en person i butiken
//Pre:  Ture
//Post: En person har laggts till i affären
void kommerIn(Affare & shoppen);

//Skickar någon till kön
//Pre:  True
//Post: En person från affären står nu i kö
void gaTillKo(Affare & shoppen);

//Visar alla kunder som inte står i kö
//Pre:  True
//Post: Kunder i butiken har skrivits ut
void visaKunder(Affare & shoppe);

//Öppnar en ny kassa
//Pre:  True
//Post: En ny kassa med tillhörande kö har öppnats
void oppnaKassa(Affare & shoppen);

//Visar en kassa kö
//Pre:  True
//Post: Skrivit ut den kön användaren valt
void tittaKo(Affare & shoppen);

//Stänger en kassa
//Pre:  True
//Post: En kassa har stängts och alla dess kunder har placerats
//      i de andra kö-erna!
void stangKassa(Affare & shoppen);

//Betjänar en kund i kassan
//Pre:  True
//Post: En kund har blivit betjänad och lämnat kassakön
void betjana(Affare & shoppen);

#endif
```

list.cpp

```
#include <iostream>

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

  head = 0;
  sizen = 0;

  std::cerr << "LIST NKONST";

}
```

```
template <class T>
List<T>::List(const List<T>& orginal)
{

  head = orginal.head;
  sizen = orginal.sizen;

}


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

  int stor;

  std::cerr << "LIST DEST";

  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;


}
```

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();

  //
  //
  //
  List(const List<T>& orginal);

  //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
```

main.cpp

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

int printMenu()
```

```
{

  using namespace std;

  cout << "   Meny   " << endl
       << "----------" << endl << endl
       << " 1. Någon kliver in" << endl
       << " 2. Någon går till kassan" << endl
       << " 3. Någon betjänas" << endl
       << " 4. En ny kassa öppnas" << endl
       << " 5. En kassa stängs" << endl
       << " 6. Titta på kassorna" << endl
       << " 7. Mingla med kunderna" << endl
       << " 0. Stäng affären" << endl
       << endl << "Ange ditt val: ";

  return mataInInt();

}

int main()
{

  Affare affaren;
  bool exit = false;

  std::cout << std::endl
            << "Välkommen in till den virtuella butiken" << std::endl
            << "---------------------------------------" << std::endl
            << std::endl
            << "\t Skriven av: Henrik Bäck" << std::endl
            << "\t             Mathias Andersson" << std::endl
            << std::endl;

  do
    {
      exit = false;
      switch(printMenu())
        {
        case 1: kommerIn(affaren);break;
        case 2: gaTillKo(affaren);break;
        case 3: betjana(affaren); break;
        case 4: oppnaKassa(affaren); break;
        case 5: stangKassa(affaren); break;
        case 6: tittaKo(affaren); break;
        case 7: visaKunder(affaren);break;
        case 0: exit = true; break;
        default: std::cout << std::endl << "Vakten säger 'Felaktigt val'" <<
std::endl;
        };
    }while(!exit);


  return 0;

}
```

Node.h

```
template <typename T>
class List;


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

Person.cpp

```
#include "Person.h"
#include <cstring>

Person::Person()
{
        name = 0;
        age = -1;
}

Person::Person(char* _name, int _age)
{
        name = new char[std::strlen(_name) + 1];
        std::strcpy(name, _name);
        age = _age;
}

void Person::copy(const Person& p)
{
        name = new char[std::strlen(p.name) + 1];
        std::strcpy(name, p.name);
        age = p.age;
}

void Person::freeMemory()
{
        if(name != 0)
        {
                delete[] name;
                name = 0;
        }
}

Person::Person(const Person& p)
{
        copy(p);
}

Person::~Person()
{
        freeMemory();
```

```
}

Person& Person::operator=(const Person& rhs)
{
        if (&rhs != this)
        {
                freeMemory();
                copy(rhs);
        }
        return *this;
}

const char* const Person::getName() const
{
        return name;
}

const int Person::getAge() const
{
        return age;
}
```

Person.h

```
#if !defined(PERSON_H___INCLUDED_)
#define PERSON_H___INCLUDED_

/* The class Person is a simple model of a real person. */

class Person
{
        public:
                //Pre: True
                //Post: A new instance of Person whose name is undefined and age is
-1 is created.
                Person();

                //Pre: (_name != 0) && (0 < _age < 120)
                //Post: A new instance of Person, whose name and age are set by the
parameters, is created.
                Person(char* _name, int _age);

                //Pre: True
                //Post: A new instance of Person is created as a copy of p. The
objects differ only in identity.
                Person(const Person& p);

                //Pre: True
                //Post: An instance of Person has seized to exist.
                ~Person();

                //Pre: True
                //Post: The right hand side object rhs has been assigned to this
object.
                //             The objects differ only in identity. This object is
returned.
```

```
                Person& operator=(const Person& rhs);

                //Pre: True
                //Post: result = the name of the Person object.
                const char* const getName() const;

                //Pre: True
                //Post: result = the age of the Person object.
                const int getAge() const;
        private:
                char* name; //Invariant: is either a name or NULL
                int age;        //Invariant: if name is NULL then age is -1,
otherwise a positive integer.

                //Pre: This object must differ in identity from p.
                //Post: This object is a copy of p, they differ only in identity.
                void copy(const Person& p);

                //Pre: True
                //Post: The dynamically allocated memory occupied by this object,
if any, is returned to the system.
                void freeMemory();
};


#endif // !defined(PERSON_H___INCLUDED_)
```

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++;
    }
}
```

queue.h

```
#ifndef __QUEUE_H__
#define __QUEUE_H__

#include "List.h"

template <class T>
class Queue
{
 public:

  //Creates a new queue
  //Pre:  True
  //Post: Has created a new queue
  Queue();
```

```
  //Destroys a queue
  //Pre:  !isEmpty()
  //Post: The queue has been destroyed
  ~Queue();

  //Places an element in the queue
  //Pre:  True
  //Post: The element, data, has been enqueued
  void enqueue(T data);

  //Removes a element from the queue
  //Pre:  !isEmpty()
  //Post: The element has been removed
  void dequeue();

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

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

  //Returns the number of elements in the queue
  //Pre:  True
  //Post: Has returned the number of elemements in queue
  int getSize();

  //Gets the entire queue
  //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
```

makefile

```
#MakeFile för affären
affar: main.o drivers.o Person.o affar.o
        g++ -o affar main.o drivers.o Person.o affar.o

main.o: main.cpp
        g++ -c main.cpp

drivers.o: drivers.cpp drivers.h
        g++ -c drivers.cpp

Person.o: Person.cpp Person.h
        g++ -c Person.cpp
```

```
affar.o: affar.cpp affar.h
	g++ -c affar.cpp
```