UPPGIFT 1 & 2
lask_klar.cpp

```
#include <iostream>
#include <iomanip>
#include <cctype>
#include <fstream>
using namespace std;

const int ANT_TKN = 100;
const int MAX_LASK = 30;
const char FILNAMN[] = "lask.dat";

struct lask
{
  char namn[ANT_TKN];
  char land[ANT_TKN];
  float pris;
  int mangd;

};

//Sparar hela katalogen till fil.
//Pre: ant >= 0, AllaSorter[] har ant antal poster.
//Post: Skrivit ant antal poster i AllaSorter till fil FILNAMN.
void skrivTillFil(const lask AllaSorter[], int ant)
{
  ofstream skrivFil(FILNAMN,ios::out);
    if(!skrivFil)
      cerr << "Det gick inte öppna filen för att spara data";
    else
      {

        for(int i = 0; i<ant; i++)
          {

            skrivFil.write(reinterpret_cast<const char * > (&AllaSorter[i]),
                           sizeof(lask));

          }

        skrivFil.close();

      }
}
```

```
//Läser in hela katalogen från fil
//Pre: ant >= 0, AllaSorter har MAX_LASK lediga platser
//Post: AllaSorter innehåller inlästa poster från fil och ant antalet inlästa
poster
void lasFranFile(lask AllaSorter[], int &ant)
{

  int i = 0;

  ifstream lasFil(FILNAMN,ios::in);
    if(!lasFil)
      cerr << "Det gick inte öppna filen för att öppna data";
    else
      {
        while(lasFil.peek()!= EOF)
          {
            lasFil.read(reinterpret_cast<char * > (&AllaSorter[i]), sizeof(lask));
            i++;
            lasFil.seekg((i)*sizeof(lask));

          }
        lasFil.close();
        ant = i;
      }

}

//Byter plats på element
//Pre: element1 och element2 existerar
//Post: element1 och element2 har bytt plats
void swap(lask& element1, lask& element2)
{
        lask temp = element1;
        element1 = element2;
        element2 = temp;
}

//Sorterar läsklistan enligt namn
//Pre: lask innehåller storlek antal platser
//Post: lask har blivit sorterad efter namn
void namnSort(lask vektor[], const int storlek)
{

  int m;

  for (int i = 0; i < storlek - 1; i++)
    for (int j = 0; j < storlek - 1; j++)
      {
      m = 0;
      while(toupper(vektor[j].namn[m]) == toupper(vektor[j+1].namn[m]))
        {
          m ++;
        }
      if(toupper(vektor[j].namn[m]) > toupper(vektor[j+1].namn[m]))
        swap(vektor[j], vektor[j+1]);
      }
 }
```

```
//Sorterar lask efter pris
//Pre: vektor[] har storlek antal poster
//Post: Sorterat vecktor[] efter vektor[].pris stigande
void prisSort(lask vektor[], const int storlek)
{
  for (int i = 0; i < storlek - 1; i++)
    for (int j = 0; j < storlek - 1; j++)
      if (vektor[j].pris > vektor[j + 1].pris)
        swap(vektor[j], vektor[j + 1]);
}

//Ser till att inmatat värde endast kan anta siffra
//Pre: True
//Post: Returnerat integer
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;
}

//Lägger till Läsk i vektor
//Pre: Vektor har ledig plats och ant >= 0
//Post: Läsk har adderats till vektor
void lagg_till(lask AllaSorter[], int & ant)
{

  if(ant < MAX_LASK)
    {
      cout << "LÄGG TILL NY SORT" << endl
           << "=================" << endl << endl
           << "Ange sortens namn: ";

      cin >> AllaSorter[ant].namn;

      cout << "Ange ursprungsland: ";

      cin >> AllaSorter[ant].land;

      cout << "Ange sortens pris (kr): ";

      cin.get();

      AllaSorter[ant].pris = mataIn();

      while(AllaSorter[ant].pris <= 0)
```

```
        {
          cout << "Felaktigt värde! Försök igen: " ;
          AllaSorter[ant].pris = mataIn();
        }

      cout << "Ange sortens mängd (cl): ";

      AllaSorter[ant].mangd = mataIn();

      while(AllaSorter[ant].mangd <= 0)
        {
          cout << "Felaktigt värde! Försök igen: " ;
          AllaSorter[ant].mangd = mataIn();
        }

      cout << endl << "Sorten lagrad";

      ant ++;
    }
  else
    cout << "Du kan inte lägga till fler artiklar!";

}

//Visar nummer och namn för all läsk i lista
//Pre: Det finns antal ant poster i AllaSorter
//Post: Skrivit ut lista på samtliga läsker
void lista_sort(lask AllaSorter[], int ant, bool sortName)
{

  cout << "LISTA ALLA LÄSKSORTER" << endl
       << "====================" << endl;

  cout << setiosflags(ios::left) << setw(10) << "Nummer" << setw(20) << "Namn"
       << setw(6) << "Pris" << endl;

    if(ant <= 0)
      cout << endl  << " ** Det finns inga artiklar i listan ** ";
    else
      {

        if(sortName == true)
          namnSort(AllaSorter, ant);
        else
          prisSort(AllaSorter, ant);

        for(int i = 0; i<ant; i++)
          cout << setw(10) << i << setw(20) << AllaSorter[i].namn << setw(6)
               << AllaSorter[i].pris << endl;
      }

    cout << endl << endl;

}
```

```
//Skriver ut information om specefik läsk
//Pre: Det finns poster i AllaSorter
//Post: Har skrivit ut information om läsk.
void skriv_info(lask AllaSorter[], int antal)
{

  int id;

  cout << "SKRIV UT INFORMATION OM LÄSK" << endl
       << "============================" << endl << endl;

  if(antal > 0)
    {
      cout << "Ange läskens ID-nummer [ange -1 för att avbryta]: ";

       id = mataIn();

       while((id < -1) || (id > antal))
        {
          cout << "Detta värde är ej giligt! Ange nytt: ";
          id = mataIn();
        }

      if(id != -1)
        {
          cout << "**********************************" << endl << endl;

          cout << "Namn: " << AllaSorter[id].namn << endl
               << "Ursprungsland: " << AllaSorter[id].land << endl
               << "Pris (kr): " << AllaSorter[id].pris << endl
               << "Mängd (cl): " << AllaSorter[id].mangd << endl;

          cout << endl << endl;
        }
    }
  else
    cout << "** Det finns inga artiklar i listan **" << endl;

}

//Kontrollerar om användaren vill avsluta
//Pre: True
//Post: Returnerat true om användaren vill avsluta, annars false.
bool kolla_avsluta()
{
  char stop;
  bool sluta = false;
  cout << "Vill du avsluta programmet? (J/N)  ";
  cin >> stop;
  if(toupper(stop) == 'J')
    sluta = true;

  cin.get();

  return sluta;
}
```

```
//Skriver ut programmets meny samt tar emot val
//Pre: True
//Post: Returnerat användarens val
int skriv_meny(bool namnSort)
{

  int val;
  cout << endl;
  cout << "LÄSKPROGRAMMET" << endl
       << "==============" << endl << endl
       << "1. Lägg till läsk" << endl
       << "2. Lista alla läsksorter" << endl
       << "3.  Byt sorteringsmetod för lista - aktuell inställning: ";

  if(namnSort)
    cout << "Namnsortering." << endl;
  else
    cout << "Prissortering." << endl;

  cout << "4. Skriv ut information om läsk (id krävs)" << endl
       << "0. Avsluta" << endl << endl
       << "Ange val: ";

  val = mataIn();

  return val;

}
```

```
int main()
{

  lask sorter[MAX_LASK];
  int antal_lagrade = 0;
  bool avsluta = false;
  int meny_val;
  bool namnsortering = true;
  int spara = 0;

  lasFranFile(sorter, antal_lagrade);

do
  {
    meny_val = skriv_meny(namnsortering);

    switch(meny_val)
      {
      case 1: lagg_till(sorter, antal_lagrade); break;
      case 2: lista_sort(sorter, antal_lagrade, namnsortering); break;
      case 3: if(namnsortering == false)
                 namnsortering = true;
              else
                 namnsortering = false;
              break;
      case 4: skriv_info(sorter, antal_lagrade); break;
      case 0: avsluta = kolla_avsluta(); break;
      default: cout << "Felaktig inmatning" << endl;
      }
  }while(avsluta != true);

do
  {
 cout << "Vill du spara dina ändringar?\nAnge: 1 för JA och 0 för NEJ" << endl
      << " : ";
 spara = mataIn();
   }while((spara > 1) || (spara < 0));

 if(spara == 1)
   skrivTillFil(sorter, antal_lagrade);

  return 0;

}
```

UPPGIFT3
Beskrivning

Koden sorterar enligt Merge sort. En vektor innehåller osorterade tal. Först delas vektorn in i mindre delar. Sedan sorteras de mindre delarna var för sig och sätts ihop med de andra mindre delarna. Slutligen sätts vektorn samman med hjälp av de mindre delarna till en sorterad vektor som motsvarar den ursprungliga.

I exemplets funkt2() så delas vektorn upp i mindre delar rekursivt. När delning inte är möjlig längre så startar sorteringen av de mindre delarna. Detta sker i funkt1().

Till funkt1() skickas vektorn som skall sorteras samt övre- och nedre gräns och ett medeltal. Dessa används för att begränsa sorteringen till just det intervallet i vektorn.

funkt1() flyttar det mittental den blivit tilldelat av funkt2() ett steg åt höger för att bilda två stycken intervall, ett till vänster och ett till höger. Detta för att funkt1() vill jämföra den första siffran i det vänstra intervallet med första siffran i det högra intervallet.

Den första for-loopen jämför den första siffran i det vänstra intervallet med den första siffran i det högra intervallet. Om talet i det vänstra intervallet är mindre än talet i det högra intervallet läggs det vänstra intervallets tal in i en temporär vektor på sin nya plats. Därefter ökas det vänstra intervallets nedre gräns ett steg. En ny jämförelse görs med de nya nedre gränsvärdena. Ifall det högra gränsvärdet är mindre än det vänstra så läggs detta in på sin nya plats och det högra gränsvärdets nedre gräns ökas med ett steg.

Därefter, i den andra och tredje for-loopen, kopierar funktionen över resterande värden från det vänstra intervallet till den temporära vektorn. Samma sak gäller för det högra intervallet, detta kopieras också över till den temporära vektorn i fall detta är nödvändigt.

I den fjärde och sista for-loopen så kopieras de sorterade värdena tillbaka till den ursprungliga vektorn där de läggs på samma plats som de har blivit tilldelade i den temporära vektorn. Detta innebär att värden kopieras tillbaka på sin rätta plats.

funkt1() körs flera gånger med olika intervall.

EXEMPEL

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

const int MAX_SIZE = 8;


int main()
{

  int vek[MAX_SIZE];

  cout << ”Mata in ” << MAX_SIZE
       << ” stycken heltal, avsluta varje tal med [RETUR].”;

  for(int i = 0; i < MAX_SIZE; i++)
      cin >> vek[i];

  funkt2(vek,0, MAX_SIZE – 1);

  for(int i = 0; i < MAX_SIZE; i++)
      cout << vek[i] << “,”;

  return 0;

}
```

Samsortering (Merge Sort)