Høgskolen i Gjøvik

Avdeling for Teknologi

KONTINUASJONSEKSAMEN

FAGNAVN:	**Grunnleggende datakunnskap og programmering**
FAGNUMMER:	**L 182 A**
EKSAMENSDATO:	**17. august 2001**
KLASSE:	**00HINDA / 00HINDB / 00HINEA 00HDMUA / 00HDMUB / 00HINGA**
TID:	**09.00-13.00**
FAGLÆRER:	**Frode Haug**
ANTALL SIDER UTLEVERT:	**7 (inkludert denne forside)**
TILLATTE HJELPEMIDLER:	**Kun læreboka ”OOP in C++”**

- Kontroller at alle oppgavearkene er tilstede.
- Innføring med penn, evt. trykkblyant som gir gjennomslag.
  Pass på at du ikke skriver på mer enn ett innføringsark om gangen (da det blir uleselige gjennomslag om flere ark ligger oppå hverandre når du skriver).
- Ved innlevering skilles hvit og gul besvarelse og legges i hvert sitt omslag.
  Oppgavetekst, kladd og blå kopi beholder kandidaten.
- Ikke skriv noe av din besvarelse på oppgavearkene.
- Husk kandidatnummer på alle ark.

Dette eksamenssettet består av tre ulike oppgavetyper:

Oppgave 1 omhandler teori fra datateknikk-delen.
Oppgave 2 omhandler hva som blir utskriften fra noen ulike program.
Oppgave 3 omhandler et litt større programmerings-case.

NB: De tre oppgavene er totalt uavhengige og kan derfor løses separat.

Oppgave 1 (30 %)

a) Internet:

a1) Forklar kort (max. ½ A4-ark) de mest typiske funksjonene/mulighetene i en browser.

a2) Hvilke tre metoder for mottak, (videre)sending og oppbevaring av email har vi ?

a3) Hva står forkortelsen "FTP" for ? Forklar kort (max. ½ A4-ark) hva "FTP" er.

b) Datamaskinen (periferiutstyr / innmat):

b1) Forklar kort (max. ½ A4-ark) om portene på en datamaskin.

b2) Redegjør kort (max. fire setninger pr.punkt) omkring:

b2-a) USB

b2-b) Cluster

b2-c) FAT

c) Datasikkerhet:

c1) Angi fem områder/typer for sårbarhet.

c2) Nevn fem karakteristika på datavirus.

c3) Nevn fem eksempler på *smitteveier* for datavirus (hvor virusene er/ligger rent fysisk).

d) Personvern / etikk:

d1) Angi fem nøkkelord for etiske verdier.

d2) Nevn fem kategorier for etiske problem*områder* ifm. informasjonsteknologi (IT).

Oppgave 2 (20 %)

a) Hva blir utskriften fra følgende program (litt hjelp: det blir 5 linjer):

```
#include <iostream>

using namespace std;

int main()   {
  int i=10, j;

  while (i > 0)  {
    cout << i << ":  ";
    for (j = 1;  j <  i*2;  j *= 2)
        cout << j << "  ";
    cout << '\n';
    i -= 2;
  }
  return 0;
}
```

b) Hva blir utskriften fra følgende program (litt hjelp: det blir 5 linjer):

```
#include <iostream>
#include <cstring>

using namespace std;

void funk(char txt[], int n = 1, char ch = 'O')
  {  txt[n] = ch;  }

void funk(char txt[], char ch)
  {  txt[3] = ch;  ch = 'E';  }

void funk(char txt[], char & ch, int n)
  {  ch = txt[n];  strcpy(txt, "HALLO");  }

int main()  {
  char tekst[] = "MALLA";
  char tegn = 'O';

  funk(tekst);            cout << tekst << '\n';
  funk(tekst, tegn);      cout << tekst << "  " << tegn << '\n';
  funk(tekst, 2, 'T');    cout << tekst << '\n';
  funk(tekst, tegn, 1);   cout << tekst << "  " << tegn << '\n';
  funk(tekst, 1, tegn);   cout << tekst << '\n';
  return 0;
}
```

Oppgave 3 (50 %)

NB: Les hele teksten for denne oppgaven nøye, før du begynner å besvare noe som helst. Studer vedlegget nøye, som inneholder mange viktige opplysninger som du trenger/ skal bruke. Legg spesielt merke til (og bruk) const'ene og variablene i hovedprogrammet.

Innledning

I denne oppgaven skal du lage et lite program som holder orden på serietabellen for spillet mellom ulike lag, f.eks. i fotball, ishockey, håndball eller basketball. Serietabellen ligger i utgangspunktet på en egen fil (TABELL.DTA, mer i **oppgave 3b**). Denne leses inn i en array av struct'er. Kampresultatene mellom to og to lag leses inn fortløpende. Lagenes data (antall: kamper, seire, uavgjort, tap, scorede/innslupne mål og poeng) oppdateres etter hvert leste resultat (**oppgave 3e**), serietabellen sorteres igjen (**oppgave 3d**) og den skrives for hvert innleste resultat ut til skjermen (**oppgave 3c**).
Din oppgave er å skrive resten av programmet angitt i vedlegget og denne oppgaveteksten.

Datastrukturen

I vedlegget kan du på den første siden se deklarasjoner/definisjoner av const'er, en struct og globale variable. *Dette skal være alt det du trenger av globale variable for å løse denne eksamensoppgaven !* Datastrukturen består altså av arrayen "tabellen" som er bygd opp av "Lag"-struct'er. Struct'ene f.o.m. indeks nr.1 t.o.m. nr."ant_lag" er i bruk i denne arrayen.

Oppgaven

a) Lag funksjonen "int finn_lag(char nvn[])".

Funksjonen leter i arrayen "tabellen" etter laget med navn lik "nvn". Om den finner et slikt lag, så returneres dets indeks i arrayen. Ellers returneres 0 (null).

Hint: Bruk "strcmp(str1, str2)" som returnerer 0 (null) når tekstene er like (dvs. 'null' forskjell).

b) Lag funksjonen "void les_fra_fil()".

Funksjonen skal lese inn aktuell serietabell fra filen "TABELL.DTA". Først på filen ligger et tall som angir hvor mange lag/linjer som videre kommer på filen. Dette tallet angir dermed "ant_lag".
Om dette tallet er for stort, skal det komme en melding, og filens videre innhold leses ikke inn.
Resten av filen består av "ant_lag" linjer, der hver linje inneholder *alle* aktuelle data om *ett* lag.

Hver linje inneholder først lagets navn (posisjon 1-19). Deretter kommer det (f.o.m. posisjon 20) syv int'er etter hverandre. Disse representerer fortløpende de tilsvarende int'ene i Lag-struct'en. Alle int'ene er adskilt med (ett eller flere) blanke tegn.

(Vi forutsetter at de virkelig *er* eksakt "ant_lag" linjer på filen, og at deres format er korrekt.)

c) Lag funksjonen "void skriv_tabell()".

Funksjonen går gjennom "tabellen" og skriver ut *alle* data om *alle* lag på en pen og oversiktlig måte. Det skrives ett lag på hver linje på skjermen, og tallene vises/presenteres pent i kolonner.

Hint: Bruk bl.a. "setw(n)".

d) Lag funksjonen ”void sorter(int n)”.

Funksjonen skal forsøke å flytte laget med indeks nr.”n” i ”tabellen” høyere opp i arrayen. Dette gjør den kun dersom lag foran den (med lavere indeks) har lavere med poeng. Dette kalles ’innstikkssortering’, så lenge alle lagene foran rykker ett hakk ned, og laget smettes innimellom der det hører hjemme i arrayen.

Hint: Vha. en while-løkke kopierer man, så lenge lagene foran har dårligere med poeng enn det aktuelle laget, lag nr.”i-1” ned til plass nr.”i”. Det aktuelle laget smettes (stikkes inn) så til slutt inn på den plassen der det sist ble kopiert et lag fra.

NB: Vi sorterer *kun* etter lagenes poeng, og ignorerer totalt deres mål-differanse ! (Som vi selvsagt hadde måttet ta hensyn til i et reelt tilfelle.)

e) Lag resten av innmaten i ”main”.

Koden som du her skal skrive er det som skal gjøre resten av ’jobben’ i hovedprogrammet. Dette gjøres bl.a. ved å bruke/tilkalle funksjonene laget i oppgavene (**3a, 3c og 3d**) ovenfor.

Det som skal gjøres i ”main” er: Be om og les inn hjemmelagets navn før enn og til slutt i en while-løkke. Programmet stanser når hjemmelagsnavnet *kun* er CR/ENTER (dvs. navnet har en lengde lik 0). Sjekk så om navnet er å finne i ”tabellen” vha. ”finn_lag(….). Om det ikke finnes, kommer en melding, og nytt hjemmelagsnavn forsøkes lest. Om navnet finnes så forsøkes det å lese bortelagsnavnet. Om dette ikke finnes, så kommer en melding, og nytt *hjemmelagsnavn* forsøkes lest. Om navnet finnes, så bes det om og leses inn antall mål både hjemme- og bortelag har scoret (du trenger *ikke* å sjekke om disse er i et realistisk intervall). Deretter oppdateres begge lagenes ”ant_kamper”, ”ant_scoret” og ”ant_innsluppet”. Ut fra målene (hvem som vant eller at det endte uavgjort), så oppdateres de aktuelle lagene med ”ant_seire”, ”ant_uavgjort”, ”ant_tap” og ”ant_poeng”. Const’er inneholder poengene som gis for seier og uavgjort (tap gir 0 poeng). Arrayen sorteres igjen for ett (når et lag vant) eller begge lagene (ved uavgjort) på egnede steder i koden. Til slutt, før innlesning av nytt hjemmelagsnavn, tilkalles ”skriv_tabell()”.

NB 1: Husk å bruke de allerede definerte variablene i ”main”. (Du trenger faktisk ikke flere!)

NB 2: Et kampresultat leses her inn på en litt ’unormal’ måte, da vi eksplisitt ber om ett og ett lags navn, og deretter hvor mange mål de har scoret. Det hadde kanskje vært mer naturlig å angi det på formen: <hjemmelag> - <bortelag> <mål hjemmelag> - <mål bortelag>
Med en slik tekststreng som input til programmet hadde blitt noe mer tuklete å håndtere/ analysere kampresultatet, derfor har vi her gjort innlesningen noe enklere/annerledes.

Klargjøring og forutsetninger

- **Oppgave 3e** kan utføres uten at **oppgave 3d** er løst/gjort. ”sorter(x)” kan fritt kalles når det er aktuelt, selv om den ikke er (ferdig) programmert !
- Det er altså ikke en del av eksamensoppgaven å skrive en oppdatert serietabell tilbake til filen ”TABELL.DTA”. Dette ville jo uansett bli svært likt med ”skriv_tabell()” (**oppgave 3c**).
- Det er plass til alle dataene fra fil i datamaskinens primærhukommelse.
- Gjør dine egne forutsetninger dersom du finner oppgaveteksten upresis eller ufullstendig. Gjør i så fall rede for disse forutsetningene først i besvarelsen din.

Lykke til !

frode@haug.com

Vedlegg: Halvferdig programkode (.tpl-fil)

```
                        //  INCLUDE:
#include <iostream>             //  cin, cout
#include <fstream>              //  ifstream
#include <cstring>              //  strcmp, strlen
#include <iomanip>              //  setw

using namespace std;

                        //  CONST:
const int NVN_LEN     = 20;     //  Max. lengde for et lags navn.
const int MAX_LAG     = 30;     //  Max. antall lag på/i serietabellen.
const int SEIER       =  3;     //  Antall poeng for en seier.
const int UAVGJORT    =  1;     //  Antall poeng for en uavgjort.


                        //  STRUCT:
struct Lag  {                   //  Struct for et lag m/
  char navn[NVN_LEN];           //    dets navn,
  int ant_kamper,               //    antall kamper totalt spilt,
      ant_seire, ant_uavgjort, ant_tap, //   antall seire/uavgjort/tap,
      ant_scoret, ant_innsluppet,       //   antall mål scoret/innsluppet,
      ant_poeng;                //    antall poeng totalt.
};


                        //  DEKLARASJON AV FUNKSJONER:
int  finn_lag(char nvn[]);      //  Oppgave 3a
void les_fra_fil();             //  Oppgave 3b
void skriv_tabell();            //  Oppgave 3c
void sorter(int n);             //  Oppgave 3d


                        //  GLOBALE VARIABLE:
int ant_lag;                    //  Antall lag/linjer i serietabellen/på filen.
Lag tabellen[MAX_LAG+1];                //  Serietabellen med lagene.


                        //  HOVEDPROGRAMMET:
int main()  {
  char navn1[NVN_LEN], navn2[NVN_LEN];      //  De to lagenes navn.
  int maal1, maal2;                         //  Målene de har scoret.
  int nr1, nr2;                             //  Deres nr. i tabellen.

  les_fra_fil();                            //  Leser serietabell fra fil.
  skriv_tabell();                           //  Skriver til skjermen.

        //  Oppgave 3e: Lag resten av innmaten
  return 0;
}
```

```
                    //  DEFINISJON AV FUNKSJONER:

int finn_lag(char nvn[])  {          //  Leter etter laget med navnet "nvn" i arrayen "tabellen".

        //  Oppgave 3a:  Lag innmaten
}


void les_fra_fil()  {                //  Leser datastrukturen/serietabellen fra fil.

        //  Oppgave 3b:  Lag innmaten
}


void skriv_tabell()  {               //  Skriver serietabellen til skjermen.

        //  Oppgave 3c:  Lag innmaten
}


void sorter(int n)  {                //  Innstikkssorterer et lag (om mulig) høyere opp på serietabellen.

        //  Oppgave 3d:  Lag innmaten
}
```