Høgskolen i Gjøvik
Institutt for informatikk og medieteknikk

KONTINUASJONSEKSAMEN

FAGNAVN: **Grunnleggende programmering**

FAGNUMMER: **IMT 1031**

EKSAMENSDATO: **9. august 2006**

KLASSE(R): **05HBIND*, 05HBINFA, 05HBISA, 05HBMETEA, 05HBINE*, 05HBGEOA**

TID: **09.00-13.00**

FAGLÆRER: **Frode Haug**

ANTALL SIDER UTLEVERT: **7 (inkludert denne forside)**

TILLATTE HJELPEMIDLER: **Alle trykte og skrevne**

- Kontroller at alle oppgavearkene er til stede.
- Innføring med penn, eventuelt trykkblyant som gir gjennomslag. Pass på så du ikke skriver på mer enn ett innføringsark om gangen (da det blir uleselige gjennomslag når flere ark ligger oppå hverandre).
- Ved innlevering skilles hvit og gul besvarelse, som legges i hvert sitt omslag.
- Oppgavetekst, kladd og blå kopi beholder kandidaten til klagefristen er over.
- Ikke skriv noe av din besvarelse på oppgavearkene. Men, i oppgavetekst der du skal fylle ut svar i tegning/tabell/kurve, skal selvsagt dette innleveres sammen med hvit besvarelse.
- Husk kandidatnummer på alle ark. Ta vare på dette nummeret til sensuren faller.

Eksamenssettet består av to ulike oppgavetyper:

Oppgave 1 omhandler hva som blir utskriften fra/av to ulike programmer.
Oppgave 2 omhandler et litt større programmerings-case.

NB: Oppgavene 1a, 1b og 2 er totalt uavhengige og kan derfor løses separat.

Oppgave 1 (30 %)

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

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

char txt[] = "DILL-DALL-DOLLOM";

int main()  {
  int i, j = 15;
  for (i = 3;  i < 18;  i+=5, j-=5)
      cout << txt[i] << ' ' << txt[j] << '\n';

  i = 12;  j = 2;
  while (j < i)  {
    cout <<  txt[i] << ' ' << txt[j] << '\n';
    j += 3;  i-= j;
  }
  return 0;
}
```

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

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

char tekst[] = "BILGUMMIDEKKOVERFLATEBEHANDLING";

void skriv(char t[], int n, int m, int d)  {
  for (int i = n;  i <= m;  i+=d)
      cout << t[i] << ' ';
  cout << '\n';
}

char hent(char t[], int nr)  {
  return t[--nr];
}

void rokk(char & a, char & b)  {
  char c = a;  a = b;  b = c;
}

int main() {
  skriv(tekst, 7, 23, 5);
  cout << hent(tekst, 9) << ' ' << hent(tekst, 27) << '\n';
  cout << char(int(tekst[19])+4) << '\n';
  rokk(tekst[3], tekst[8]);    skriv(tekst, 0, 7, 1);
  rokk(tekst[4], tekst[hent(tekst, 23)-'D']);
  skriv(tekst, 1, 4, 3);
  return 0;
}
```

Oppgave 2 (70 %)

NB: Les *hele* teksten for denne oppgaven *nøye*, før du begynner å besvare noe som helst. Studer vedlegget, som inneholder mange viktige opplysninger som du trenger/skal bruke. Legg spesielt merke til const'ene (bruk dem a*ktivt*), klassen, globale variable og funksjonene `finn_person(……)` og `Person::les_dagnr()` (bruk også disse *aktivt*). Gjør dine egne forutsetninger dersom du finner oppgaveteksten upresis eller ufullstendig. Gjør i så fall rede for disse forutsetningene *først* i besvarelsen din.

I denne oppgaven skal du lage et lite program som holder orden på ulike personers *faste* fritids-aktiviteter i løpet av ukens syv dager.

Datastrukturen

Arrayen `personer` består av MAXPERS `Person`-objekter (se første siden i vedlegget). `siste_brukt` angir antallet av disse som for øyeblikket er i bruk. Vi bruker indeks nr.1 og oppover. Den to-dimensjonale char-arrayen `ukedag` inni klassen representerer på hver linje teksten for den faste fritidsaktiviteten personen har på den aktuelle ukedagen. Indeks nr.0 bruker vi ikke, så nr.1 tilsvarer søndag, nr.2 er mandag, …..., nr.7 er lørdag. Dvs. hver person kan *kun* ha *en* fast fritidsaktivitet for hver dag (litt *vel* forenklet selvsagt ift. den reelle hverdag ☺).
På den første siden i vedlegget kan du se ulike deklarasjoner/definisjoner. *Dette skal være alt du trenger av klasser, datamedlemmer og globale variable for å løse denne eksamensoppgaven!*

Oppgaven

a)

Lag funksjonene `void ny_person()` og `void Person::les_data()`

Det sjekkes først om flere personer kan legges inn. Om så *ikke* er tilfelle kommer det en melding. I motsatt fall tas et nytt objekt i bruk og dets data leses inn. Innlesningen av data foregår ved at først leses personens navn inn. Deretter blir teksten på aktiviteten for *alle* ukedagene satt til
"-----" (betyr 'tomt/ingenting'). Til slutt tilkalles bare funksjonen `ny()` (se oppgave 2d).

b)

Lag funksjonene `void skriv_person()` og `void Person::skriv_alt()`

Det spørres først om navnet til en person. Om ingen objekter har dette navnet (bruk `finn_person(……)`) kommer det en melding. I motsatt fall blir den aktuelle personen bedt om å skrive ut *alle* sine aktivitetsdata. Det skal skrives en linje pr.ukedag. Denne linjen består av ukedagens navn ("Søndag", "Mandag", …, "Lørdag") etterfulgt av teksten for denne dagens aktivitet. (Om denne er 'tom'/finnes ikke så skrives det altså ut "-----".)

c) **Lag funksjonene** `void skriv_alle_personer()` og `void Person::skriv_navn()`

For alle `Person`-objektene som er i bruk skrives kun deres navn ut på skjermen.

d) **Lag funksjonene** `void ny_aktivitet()` og `void Person::ny()`

Funksjonen `ny_aktivitet()` blir nesten identisk til `skriv_person()`.
Beskriv/bemerk endringene ift. koden du laget i oppgave 2b.
Funksjonen `ny()` tilbyr brukeren (helt til hun/han velger dag nr.0) å legge inn en tekst for ulike ukedagers aktiviteter. Bruk funksjonen `les_dagnr()`.

e) **Lag funksjonen** `void Person::fjern()`

Funksjonen nullstiller (legger inn "-----") ulike ukedagers aktiviteter til brukeren velger dag nr.0.

NB: Funksjonen `fjern_aktivitet()` skal *ikke* kodes, da den blir "identisk" til de laget i oppgavene 2b og 2d.

f) **Lag funksjonene** `void skriv_til_fil()` og `void Person::skriv_til_fil(.....)`

Filen "AKTIVITET.DTA" inneholder først *ett* tall som angir hvor mange `Person`-poster som kommer. Deretter følger N slike poster, der hver av dem har følgende format: En linje med personens navn, og deretter syv linjer med teksten (kan også være "-----") for de ulike ukedagens aktiviteter.
Det gåes altså gjennom alle `Person`-objekter som er i bruk, og hver av dem sørger for en utskrift som tilfredsstiller formatet angitt ovenfor.

g) **Lag funksjonene** `void les_fra_fil()` og `void Person::les_fra_fil (.....)`

Disse funksjonene sørger sammen for at *alle* data fra filen "AKTIVITET.DTA" blir lest inn i datastrukturen. (Formatet er beskrevet i oppgave 2f ovenfor.)

NB: Legg merke til at `Person::les_fra_fil (.....)` *ikke* tar personens navn som parameter, men skal selv lese inn dette.

Lykke til - også med ukens andre aktiviteter!

frode@haugianerne.no

Vedlegg: Halvferdig programkode (.tpl-fil)

```
                        // INCLUDE:
#include <iostream>             // cin, cout
#include <fstream>              // ifstream, fstream
#include <cctype>               // toupper
#include <cstring>              // strcpy, strcmp
using namespace std;

                        // CONST:
const int ANT_DAGER  =  7;      // Antall ukedager.
const int STRLEN     = 30;      // Max. tekstlengde.
const int MAXPERS    = 50;      // Max. antall personer i kartoteket.


                        // KLASSER:
class Person  {                 // Personen og dens daglige aktiviteter:
   private:
        char navn[STRLEN];                      // Navn.
        char ukedag[ANT_DAGER+1][STRLEN];       // Aktivitetene de ulike ukedagene.
        int  les_dagnr();                       // Privat hjelpefunksjon.

    public:                                     // Returnerer true/false til om personen har 'navn' lik 't':
        bool er_lik(char nvn[])  { return (!strcmp(navn, nvn)); }
        void les_data();                        // Deklarasjon av medlemsfunksjoner:
        void skriv_alt();                       void skriv_navn();
        void ny();                              void fjern();
        void skriv_til_fil(ostream* ut);        void les_fra_fil(istream* inn);
};

                        // DEKLARASJON AV FUNKSJONER:
void skriv_meny();          char les_kommando();        int finn_person(char t[]);
void ny_person();           void skriv_person();        void skriv_alle_personer();
void ny_aktivitet();        void fjern_aktivitet();
void skriv_til_fil();       void les_fra_fil();


                        // GLOBALE VARIABLE:
Person personer[MAXPERS+1];             // Array med Person-objekter.
int siste_brukt;                        // Siste brukte objekt hittil (nr.1 --> )


int main()  {           // HOVEDPROGRAM:
  char kommando;

  les_fra_fil();                                // Oppgave 2g

  skriv_meny();
  kommando = les_kommando();
```

```
  while (kommando != 'Q')  {
     switch (kommando)  {
         case 'P':   ny_person();              break;  //  Oppgave 2a
         case 'S':   skriv_person();           break;  //  Oppgave 2b
         case 'A':  skriv_alle_personer();     break;  //  Oppgave 2c
         case 'N':  ny_aktivitet();            break;  //  Oppgave 2d
         case 'F':   fjern_aktivitet();        break;  //  Oppgave 2e
         default:   skriv_meny();              break;
    }
    kommando = les_kommando();
  }
 skriv_til_fil();                                    //  Oppgave 2f
 return 0;
}

                          //  DEFINISJON AV KLASSE-FUNKSJONER:
int Person::les_dagnr()  {              //  Leser og returnerer et tall i intervallet 0-7:
 int nr;
 do  {
    cout << "\n\tDagnr (0=avslutt, 1=søndag, ...., 7=lørdag): ";
    cin >> nr;            cin.ignore();
 } while (nr < 0  ||  nr > ANT_DAGER);
 return nr;
}

void Person::les_data()  {              //  Leser navn og "nullstiller":
        //  Oppgave 2A:  Lag innmaten
}

void Person::skriv_alt()  {             //  Skriver alt om en person:
        //  Oppgave 2B:  Lag innmaten
}

void Person::skriv_navn()  {            //  Skriver kun personens navn:
        //  Oppgave 2C:  Lag innmaten
}

void Person::ny()  {                    //  Leser ny(e) aktivitet(er):
        //  Oppgave 2D:  Lag innmaten
}

void Person::fjern()  {                 //  Fjerner en/flere aktivitet(er):
        //  Oppgave 2E:  Lag innmaten
}

void Person::skriv_til_fil(ostream* ut)  {     //  Skriv til fil:
        //  Oppgave 2F:  Lag innmaten
}

void Person::les_fra_fil(istream* inn)  {      //  Les fra fil:
        //  Oppgave 2G:  Lag innmaten
}
```

```
                        //  DEFINISJON AV FUNKSJONER:
void skriv_meny()  {                    //  Presenterer lovlige menyvalg:
  cout << "\n\nFØLGENDE KOMMANDOER ER LOVLIG:\n";
  cout << "\tP = ny Person\n";
  cout << "\tS = Skriv person\n";
  cout << "\tA = skriv Alle personer\n";
  cout << "\tN = Ny aktivitet\n";
  cout << "\tF = Fjern aktivitet\n";
  cout << "\tQ = quit/avslutt\n";
}

char les_kommando()  {                  //  Henter ett ikke-blankt upcaset tegn:
  char ch;
  cout << "\n\nOppgi ønske:  ";
  cin >> ch;   cin.ignore();
  return toupper(ch);
}

int finn_person(char t[])  {            //  Returnerer (om mulig) indeksen for personen
  for (int i = 1;  i <= siste_brukt;  i++)   //  med navn lik 't':
     if (personer[i].er_lik(t)) return i;
  return 0;
}

void ny_person()  {                     //  Ny person i kartoteket:
        //  Oppgave 2A:  Lag innmaten
}

void skriv_person()  {                  //  Skriv alt om en gitt person:
        //  Oppgave 2B:  Lag innmaten
}

void skriv_alle_personer()  {           //  Skriver alle personenes navn:
        //  Oppgave 2C:  Lag innmaten
}

void ny_aktivitet()  {                  //  Legger inn ny(e) aktivitet(er):
        //  Oppgave 2D:  Lag innmaten
}

void fjern_aktivitet()  {               //  Fjerner aktivitet(er):
        //  Innmaten "identisk" til oppgave 2B og 2D
}

void skriv_til_fil()  {                 //  Skriver datastrukturen til fil:
        //  Oppgave 2F:  Lag innmaten
}

void les_fra_fil()  {                   //  Leser datastrukturen fra fil:
        //  Oppgave 2G:  Lag innmaten
}
```