Høgskolen i Gjøvik
Institutt for informatikk og medieteknikk

E K S A M E N

FAGNAVN: **Grunnleggende programmering**

FAGNUMMER: **IMT 1031**

EKSAMENSDATO: **19.desember 2005**

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[] = "ABBA-BA-AB-BE-EBBA";

int main()  {
  int i, j = 8;
  for (i = 9;  i < 18;  i+=3, j-=2)
      cout << txt[i] << ' ' << txt[j] << '\n';

  i = 0;  j = 17;
  while (i < j)  {
    i += 2; j /= i;
    cout <<  txt[i] << ' ' << txt[j] << '\n';
  }
  return 0;
}
```

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

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

enum Type { Taper, Vinner };

class Lag  {
  private:
     char navn[10];
     Type type;
  public:
     Lag()            {  strcpy(navn, "Arsenal");  type = Vinner;  }
     Lag(char t[], Type tt)  {  strcpy(navn, t);  type = tt;  }
     void skriv() { cout << navn << " - "
                         << ((type==Vinner)? "Vinner" : "Taper") << '\n';  }
     char funk1()     {  return (char(navn[3]+3));  }
     bool funk2(Lag l) {  return (navn[5] == l.navn[1]);  }
     void funk3()     {  strcat(navn, "UU-UUU");  cout << navn << '\n';  }
};


int main()  {
  Lag lag1, lag2("ManU", Taper);
  lag2.skriv();    lag1.skriv();
  cout << lag2.funk1() << '\n';
  if (lag1.funk2(lag2)) cout << "2-0\n"; else cout << "3-0\n";
  lag2.funk3();
  return 0;
}
```

Oppgave 2 (70 %)

NB1: Oppgavene 2a, 2b, 2d og 2e teller 10% hver, oppgavene 2c og 2f teller 15% hver.

NB2: 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, global variabel og funksjonen `int les(char t[], int min, int max)`**. Husk å bruke denne funksjonen *aktivt*, dvs. *alle* steder der det er aktuelt med innlesning av et tall.**

I denne oppgaven skal du lage et lite program som holder orden på bestilling/reservering/booking av et låneobjekt (f.eks. en idrettshall o.l.) til ulike klokkeslett (på timebasis) i løpet av en syv dagers periode.

Datastrukturen

En ukedag er representert vha. klassen `Ukedag`. Det finnes kun ANT_DAGER slike objekter, representert vha. arrayen `uka`. I denne er indeksene 0 (mandag) til ANT_DAGER-1 (søndag) i bruk. Klassen inneholder kun to datamedlemmer: `navn` og den to-dimensjonale char-arrayen `book`. Denne siste arrayen inneholder navnene (på hver linje) for den personen som har bestilt/reservert/ booket låneobjektet de ulike klokkeslettene i løpet av en dag. Det er mulig å bestille fra kl.07.00 til kl.21.00 hver dag (dvs. siste utlånstime starter kl.20.00). Linje nr.0 tilsvarer kl.07.00, mens linje nr. ANT_TIMER-1 tilsvarer kl.20.00.
På den første siden i vedlegget kan du se deklarasjoner/definisjoner av const'er, klassen og global variabel. *Dette skal være alt du trenger av klasser, datamedlemmer og globale variable for å løse denne eksamensoppgaven!*

Oppgaven

a)

Lag funksjonene `void skriv_dag()` og `void Ukedag::skriv()`

Det spørres først etter et nummer (1-7) for en ukedag. Denne ukedagen skriver så ut *alle* sine data på skjermen, dvs. sitt eget navn og navnene til de personene som har bestilt til de ulike klokkeslettene (kl.07.00-20.00) – alt dette på hver sin linje. Ledige klokkeslett/timer skal også skrives ut.

b)

Lag funksjonene `void resett_dag()` og `void Ukedag::resett()`

Det spørres først etter et nummer (0-7) for en ukedag. 0 betyr ingen ukedag (man "angrer" liksom på å ha startet funksjonen). Dersom valget er 1-7 så skal den aktuelle ukedagen slette/resette alle sine bestillinger, dvs. inn i arrayen `book` legges det inn en tom tekst ("") på hver eneste linje.

c)

Lag funksjonene `void bestill_time()` og `void Ukedag::bestill()`

Det spørres først etter et nummer (0-7) for en ukedag. 0 betyr ingen ukedag (man "angrer" liksom på å ha startet funksjonen). Dersom valget er 1-7 så skal den aktuelle ukedagen be om et klokkeslett i intervallet 7-20. *Om* den aktuelle timen er ledig, så leses og legges inn bestillerens navn. I motsatt fall kommer det en melding om at timen/klokkeslettet er allerede opptatt.
NB: Husk at klokkeslettet 7-20 må "konverteres" til indeksene 0 - ANT_TIMER-1.

d) **Lag funksjonene** `void avbestill_time()` og `void Ukedag::avbestill()`

Det spørres først etter et nummer (0-7) for en ukedag. 0 betyr ingen ukedag (man "angrer" liksom på å ha startet funksjonen). Dersom valget er 1-7 så skal den aktuelle ukedagen be om et klokkeslett i intervallet 7-20. *Om* den aktuelle timen er bestilt/reservert/booket så skal den settes ledig igjen (kopiere inn en tom tekst ("") på linjen). I motsatt fall kommer det en melding om at timen/klokkeslettet allerede er ledig.

e) **Lag funksjonene** `void skriv_til_fil()` og `void Ukedag::skriv_til_fil(……)`

Filen "BOOKING.DTA" inneholder 15 linjer for hver av ukens syv dager. På den første av disse 15 linjene ligger alene dagens navn (forkortet til "Man", "Tirs", …., "Søn"). De 14 neste linjene for *en* dag inneholder data om hvem som har booket til de ulike klokkeslettene. På hver slik linje ligger først tegnet '+' eller '-'. '+' etterfølges av *ett* blankt tegn og så bestillerens navn på resten av linja. '-' betyr at timen er ledig, dvs. ingen bestilling. Denne har ingenting etter seg på linja. Funksjonene skal åpne fila for utskrift, gå gjennom *alle* ukedagene, og hvert av dem skriver sitt innhold ut til fila etter formatet beskrevet ovenfor.

f) **Lag funksjonene** `void les_fra_fil()` og `void Ukedag::les_fra_fil (……)`

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

Klargjøring og forutsetninger

- Det er *kun* mulig å bestille for *en* time av gangen. Bestilling av dobbelttimer må altså utføres ved å kjøre kommandoen 'B' to ganger rett etter hverandre.
- Programmet håndterer *kun en* uke (seks dager frem, pluss dagen i dag). Dette medfører bl.a. at:
  - det *ikke* er mulig å legge inn bestillinger på faste ukedager og tidspunkt i ukesvis fremover.
  - om dagen i dag f.eks. er torsdag (indeks nr.3 i `uka`), så vil indeks nr.4-6 representere fredag-søndag *inneværende* uka, mens indeks nr.0-2 vil være mandag-onsdag *neste* uke.
  - når programmet startes for dagen (vi forutsetter at maskinen er skrudd av om natta), så spørres det etter hvilket dagnummer som skal resettes. Dette er gårsdagen - dagen før enn den nye nåværende dag. Det er viktig at den blir resatt, så lenge den i dag nå skal representere den dagen som er lengst frem i tid (om seks dager).
- 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 - og ha en god dag/uke !
frode@haugianerne.no

Vedlegg: Halvferdig programkode (.tpl-fil)

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

                        //  CONST:
const int ANT_DAGER  =  7;      //  Antall dager i en uke.
const int ANT_TIMER  = 14;      //  Timene i løpet av en dag (kl.07.00-20.00).
const int START_TIME =  7;      //  Startklokkeslett om morgenen.
const int STRLEN1    =  5;      //  Max. tekstlengde for en ukedags navn.
const int STRLEN2    = 30;      //  Max. tekstlengde for en bestillers navn.

                        //  KLASSER:
class Ukedag  {                 //  Ukedag for bestilling/booking:
  private:
    char navn[STRLEN1];         //  Navn.
    char book[ANT_TIMER][STRLEN2];  //  Navn på bestiller et gitt klokkeslett.

  public:
    void skriv();               //  Deklarasjon av medlemsfunksjoner:
    void resett();
    void bestill();
    void avbestill();
    void skriv_til_fil(ostream* ut);
    void les_fra_fil(char nvn[], istream* inn);
};

                        //  DEKLARASJON AV FUNKSJONER:
void skriv_meny();
char les_kommando();
int  les(char t[], int min, int max);
void skriv_dag();
void resett_dag();
void bestill_time();
void avbestill_time();
void skriv_til_fil();
void les_fra_fil();


                        //  GLOBALE VARIABLE:
Ukedag uka[ANT_DAGER];          //  Array med Ukedag-objekter.
```

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

  les_fra_fil();                                    //  Oppgave 2f
  resett_dag();                                     //  Oppgave 2b

  skriv_meny();
  kommando = les_kommando();
  while (kommando != 'Q')  {
     switch (kommando)  {
        case 'S':  skriv_dag();          break;    //  Oppgave 2a
        case 'B':  bestill_time();       break;    //  Oppgave 2c
        case 'A':  avbestill_time();     break;    //  Oppgave 2d
        default:   skriv_meny();         break;
     }
     skriv_til_fil();                               //  Oppgave 2e
     kommando = les_kommando();
  }
 cout << "\n\n";
 return 0;
}

                          //  DEFINISJON AV KLASSE-FUNKSJONER:
void Ukedag::skriv()  {               //  Skriver ALLE data om en ukedag:
        //  Oppgave 2A:  Lag innmaten
}

void Ukedag::resett()  {              //  Sletter ALLE bestillinger:
        //  Oppgave 2B:  Lag innmaten
}

void Ukedag::bestill()  {             //  Foretar EN bestilling:
        //  Oppgave 2C:  Lag innmaten
}

void Ukedag::avbestill()  {           //  Fjerner/sletter EN bestilling:
        //  Oppgave 2D:  Lag innmaten
}

void Ukedag::skriv_til_fil(ostream* ut)  {                    //  Skriver til fil:
        //  Oppgave 2E:  Lag innmaten
}

void Ukedag::les_fra_fil(char nvn[], istream* inn)  {         //  Leser fra fil:
        //  Oppgave 2F:  Lag innmaten
}
```

```
                        //  DEFINISJON AV FUNKSJONER:
void skriv_meny()  {                    //  Presenterer lovlige menyvalg:
   cout << "\n\nFØLGENDE KOMMANDOER ER LOVLIG:\n";
   cout << "\tS = Skriv alle bestillinger en gitt dag\n";
   cout << "\tB = Bestill/reserver/book en time\n";
   cout << "\tA = Avbestill en time\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 les(char t[], int min, int max)  {   //  Leser et tall i et visst intervall.
   int n;
   do  {                                //  Skriver ledetekst:
      cout << '\t' << t << " (" << min << '-' << max << "): ";
      cin >> n;                         //  Leser inn ett tall.
   } while(n < min  ||  n > max);       //  Sjekker at i lovlig intervall.
  cin.ignore();                         //  Forkaster ett tegn ('\n').
  return n;                             //  Returnerer innlest tall.
}

void skriv_dag()  {                     //  Skriv ALLE bestillinger EN gitt dag:
        //  Oppgave 2A:  Lag innmaten
}

void resett_dag()  {                    //  Slett ALT en gitt dag:
        //  Oppgave 2B:  Lag innmaten
}

void bestill_time()  {                  //  Bestill/reserver/book en time:
        //  Oppgave 2C:  Lag innmaten
}

void avbestill_time()  {                //  Avbestill en time:
        //  Oppgave 2D:  Lag innmaten
}

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

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