DATO: 26 februar 1997

HØGSKOLEN I STAVANGER
AVDELING FOR TEKNISK-NATURVITENSKAPELIGE FAG
EKSAMEN I: TE495 Programmering 2
LENGDE PÅ EKSAMEN: 5 timer, 09.00-14.00
TILLATTE HJELPEMIDLER: Alle skrevne; kalkulator
OPPGAVESETTET BESTÅR AV: 4 sider

Råd for oppgaveløsning:

Dersom du finner at teksten ikke gir deg tilstrekkelig informasjon, kan du gjøre dine egne forutsetninger. Du må i så fall presisere disse forutsetningene i besvarelsen.

Vær kort og konsis i besvarelsene. Dersom du får knapp tid er det bedre å skrive pseudokode/skisser for de oppgaver du ikke får tid til fremfor å la oppgavene være blanke.

Ved besvarelse av programmeringsoppgavene er det vesentlig at du er nøye med dokumentasjonen. *Kommenter rikelig!*

Oppgave 1

(a) [2%] Et C++-program deles gjerne opp i flere kildefiler. Hvorfor?

(b) [2%] Hva er forskjellen mellom `#include <fil.h>` og `#include "fil.h"` ?

(c) [2%] Hvordan overføres en *tabell* som parameter til en funksjon i C++? F.eks., hva skjer ved følgende funksjonskall:

```
int tab[10];
funk(tab);
```

(d) [2%] Skriv en funksjon som antydet ovenfor, hvor vi antar at funksjonen kun skal nullstille alle elementene i en 10-elements tabell.

(e) [2%] Hvordan ville du teste om en heltallsvariabel `n` var delelig med 17?

(f) [2%] Skriv en funksjon som kan konvertere grader Fahrenheit til grader Celsius. Konverteringen følger denne formelen:

$$^\circ\mathbf{C} = 5/9 \cdot (^\circ\mathbf{F} - 32)$$

(g) [2%] C++ gir oss mulighet til å lage funksjoner med *standard* (default) argumenter. Disse må imidlertid gis som de *siste* parametrene til funksjonen. Hvorfor?

(h) [2%] Skriv en funksjon `int Ordnet(int p,int q,int r)` som returnerer `1` dersom argumentene er i ordnet rekkefølge, dvs. $p \leq q \leq r$, og `0` ellers.

(i) [2%] Forenkle linjene nedenfor så mye du kan. Bli kvitt `switch`-setningen.

```
switch (hoyde>1.85)
{
  case 0:
    break;
  default:
    cout << "Personen er høy.\n";
}
```

(j) [2%] Hva i all verden er dette?

```
void X(char*a,char*b){while(*a++=*b++);}
```

Oppgave 2

En matematisk enhet er et *sett*, dvs. en beholder for en samling av relaterte elementer. Samlingen inneholder kun opplysninger om et element er med i settet eller ikke. Vi kan for eksempel tenke oss et sett av heltall:

$$\mathbf{S} = \{1, 2, 3, 5, 7\}$$

Dette settet inneholder 5 tall, og f.eks. tallet 3 *er* i settet, mens tallet 4 er *ikke* i settet.

Vi kan definere operasjoner på sett som f.eks.:

- Innsetting: et nytt tall kan legges inn i et sett. Hvis tallet allerede finnes i settet, skjer ingenting.
- Uttak: et tall kan fjernes fra et sett. Hvis tallet ikke finnes i settet, skjer ingenting.
- Test: en kan sjekke om et tall finnes i et sett.

(a) **[5%]** Vi begrenser oss her til sett sett av tallene 0 ... 99, begge inkludert. Lag en deklarasjon av en klasse `Sett` for dette. Til innsetting av elementer bruker vi `operator+=`, til uttak brukes `operator-=`, og til testing bruker vi `operator[]`.

(b) **[5%]** Skriv standard constructor for klassen, samt operatorene beskrevet ovenfor. Følgende programsnutt skal kunne bruke klassen:

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

void main()
{
  Sett S; // Settet skal være tomt til å begynne med
  S += 1;
  S += 2;
  S += 3;
  S -= 3;
  if (S[1]) cout << "S inneholder tallet 1.\n";
  if (S[3]) cout << "S inneholder tallet 3.\n";
}
```

Utskrift blir:

```
S inneholder tallet 1.
```

(c) **[10%]** Utvid klassen til å inneholde (skriv både prototyper og funksjonskropp):

- *Union*-operatorer, dvs. et sett skal kunne lages som unionen av to andre sett, og innholdet skal bli alle elementer som er med i ett eller begge settene vi lager unionen av. Bruk `operator+` og `operator+=`, slik at vi kan skrive f.eks. `S1 += S2` eller `S3 = S1+S2;` der `S1`, `S2` og `S3` alle er sett;
- *Snitt*-operatorer, som lager et sett ved kun å ta med elementer som finnes i *begge* settene vi tar snittet av. Bruk `operator*` og `operator*=`.

(d) **[5%]** Lag en utskriftsoperator som skriver ut innholdet i et sett f.eks. slik (utskrift av settet `S` i deloppgave (b)):

```
cout << "S inneholder: " << S;
```

Utskrifter blir

```
S inneholder: { 1 2 }
```

Oppgave 3

Det kan være svært så plundrete å huske hvordan utskrift av tall formateres i C++. For eksempel, hvis man vil skrive ut flyttall med to desimaler til skjermen, må man gjøre noe slik som:

```
cout.setf(ios::showpoint);
int oldprecision = cout.precision(2);
cout << tall;
cout.precision(oldprecision);
cout.unsetf(ios::showpoint);
```

En annen mulighet kan være å lage egne klasser som tar vare på sine egne formateringsinstillinger. Det skal vi gjøre her. Utgangspunktet er en abstrakt basisklasse `flyttall`:

```
class flyttall {
protected:
  float v;
public:
  flyttall(float f=0) : v(f) {};
  operator float() { return v; }
  virtual void print(ostream&) = 0;
  friend ostream& operator<<(ostream &os, flyttall &f)
    { f.print(os); return os; };
  friend istream& operator>>(istream &os, flyttall &f)
    { return os >> f.v; };
};
```

(a) **[5%]** Forklar kort de forskjellige delene av denne deklarasjonen. Hva er det som gjør klassen abstrakt, og hva innebærer det?

(b) **[10%]** Lag en avledet klasse `flyttall_2`, som ved utskrift alltid skriver tall med to desimaler. Et program som bruker en slik klasse samt utskrift gis under. (Husk at constructorer ikke kan arves.)

```
#include <iomanip.h>
#include "flyttall.h"

void main()
{
  flyttall_2 f2;
```

```
  float f;
  cout << "Gi et flyttall: ";
  cin >> f2;
  f = f2;
  cout << "float:       " << f  << endl;
  cout << "flyttall_2: " << f2 << endl;
  cout << "float:       " << f  << endl;
}
```

Utskriften skal bli:

```
Gi et flyttall: 10
float:      10
flyttall_2: 10.00
float:      10
```

(c) **[10%]** Avled en mer generell klasse `flyttall_n` som gjør at du kan sette ønsket presisjon selv.

Oppgave 4

(a) **[15%]** Lag et program som kan undersøke om to filer har identisk samme innhold. Du bestemmer selv hvordan brukeren av programmet skal skrive inn filnavnene.

Oppgave 5

(a) **[5%]** Lag en liten klasse som implementerer en tabell av 10 `int`-tall med tabellgrense-sjekking. Mao. skal følgende program kunne kjøre, med etterfølgende utskrift:

```
#include <iostream.h>
#include "tabell10.h"

void main()
{
  Tabell10 T;
  T[3] = 10;
  cout << T[3] << endl;
  T[11] = 10;
}
```

Utskrift:

```
10
Ugyldig tabellindeks. Stopp.
```

(b) **[10%]** Utvid tabellklassen slik at den kan brukes til å lage tabeller av enhver datatype, inkludert dine egne klasser (bruk et klasse-template). Skriv også et lite testprogram som viser hvordan dette skal brukes.