DATO: 23 mai 1996

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:

Les hele oppgaveteksten før du begynner å svare på de enkelte oppgavene. 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. Skriv tilstrekkelig med kommentarer.

Oppgave 1

Svar kun *ja* eller *nei* til følgende påstander:

(a) [1%] Alle C++-programmer må inneholde linjen `#include <iostream.h>`.

(b) [1%] `break` brukes til å avslutte `for`-, `while`-, `switch`- og `if`-setninger.

(c) [1%] Denne funksjonen beregner fakultet (!) for positive tall:

```
int KanskjeFakultet(int x)
{
  int p = 1;
  while (x) p *= x--;
  return p;
}
```

(d) [1%] Hvis du lager en *constructor* til en klasse, må du også lage en *destructor*.

(e) [1%] En objekt av en avledet klasse er også et objekt av basisklassen.

(f) [1%] Alle datamedlemmer i en klasse må deklareres som `private`.

(g) [1%] En tabell av pekere til flyttall kan deklareres slik: `float *p[5]`.

(h) [1%] En peker til en tabell av flyttall kan deklareres slik: `float *(p[5])`.

(i) [1%] Denne rutinen beregner absoluttverdien av et flyttall:

```
float &KanskjeAbsoluttVerdi(float &f)
{
  float a;
  a = f>=0.0 ? f : -f;
  return a;
}
```

(j) [1%] En referanse og en peker er samme ting.

(k) **[1%]** Vi kan lage vår egen utskriftsfunksjon for en klasse ved å definere en ny global funksjon `operator<<`.

(l) **[1%]** *Overloading* betyr å lage nye funksjoner med samme navn som gamle, men med forskjellig returverdi.

(m) **[1%]** Tabeller vil alltid referanseoverføres når de er argumenter til funksjoner.

(n) **[1%]** Strukturer og klasseforekomster vil alltid referanseoverføres når de er argumenter til funksjoner.

(o) **[1%]** En *abstrakt klasse* inneholder ekte virtuelle funksjoner.

Oppgave 2

Morsekode er et gammelt system for enkoding av bokstaver og tall, laget i 1832 til bruk i telegrafi. En tabell med morsekoder for tall er gitt under. Morsekodene består av to symboler, her gjengitt som "prikk" og "strek" (`.` og `-`).

Tall	1	2	3	4	5
Morsekode	`.----`	`..---`	`...--`	`....-`	`.....`
Tall	6	7	8	9	0
Morsekode	`-....`	`--...`	`---..`	`----.`	`-----`

(a) **[10%]** Skriv et program som kan lese inn tall og skrive ut den ekvivalente morsekoden. De separate kodeordene skal adskilles av et mellomrom. Kodeordene skal leses fra en tabell.

(b) **[10%]** Beskriv kort hvordan du ville løst den motsatte oppgaven, dvs. et system som leser inn prikker og streker (de enkelte symbolgruppene adskilt av mellomrom), og "dekoder" dette til normale tall. Skriv gjerne programkode dersom det letter forklaringen.

Oppgave 3

En "idiotsikker" klasse skal lages som kan behandle tekststrenger av vilkårlig lengde. Klassen skal hete `Streng`, og ha følgende brukergrensesnitt:

- Objekter skal kunne initialiseres via en `char*`, dvs. en karaktertabell.
- Utskriftsmuligheter (via `operator<<`).
- Sammenskjøting av tekstrenger via `operator+`.
- Uttak (og forandring) av enkeltelementer via `operator[]`. Her skal feilsjekking utføres, slik at ugyldig indeksering gir programavbrudd.

Programmet under skal kunne bruke klassen din:

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

```
void main()
{
  Streng A = "Her er ";
  Streng B = "en streng.";
  Streng C;
  C = A+B;
  cout << "Strengen er: " << C << endl;
  cout << "Element 8 er: " << C[8] << endl;
  C[8] = 't';
  cout << "Strengen er: " << C << endl;
  C[100] = 'x';
}
```

Utskriften fra dette programmet skal være:

```
Strengen er: Her er en streng.
Element 8 er: n
Strengen er: Her er et streng.
FEIL: Ulovlig Streng-indeks (100). Avslutter.
```

(a) **[10%]** Tenk gjennom hva klassen må inneholde, og lag en klassedeklarasjon. Kommenter koden din slik at den blir lett forståelig.

(b) **[15%]** Implementer koden ved å skrive de tilhørende funksjonene.

Oppgave 4

I denne oppgaven skal et system lages som kan brukes til å fakturere brukere av telefon. Utgangspunktet er en tekstfil som har blitt laget av en liten intern telefonsentral. Hver gang en utgående telefon har funnet sted, har dette blitt registrert med en linje i filen, hvor følgende informasjon er registrert:

- *internnummer*, dvs. nummeret det ringes fra (kun 0 til 9),
- *dato*, på formen `dd mm åå`,
- *start-tid* og *slutt-tid*, begge på formen `tt mm ss`,
- *eksternnummer*, dvs. nummeret det ringes til.

Et eksempel på en slik fil gis under.

```
1   23 05 96   09 00 00   09 01 54   51832016
2   23 05 96   10 45 16   11 04 18   006142266675
1   23 05 96   11 01 12   11 02 11   51891177
3   23 05 96   11 14 54   11 21 02   22207655
```

For å gjøre oppgaven håndterlig, begrenser vi oss til tre takstgrupper: *lokal*, dvs. oppringt nummer begynner med `51`, *utland*, dvs. oppringt nummer begynner med `00`, og *"riks"*, dvs. alt annet.

For lokalsamtaler varer et tellerskritt 3 minutter, for rikssamtaler varer det 1 minutt, og for utlandssamtaler 10 sekunder. Et tellerskritt koster 73 øre.

(a) **[10%]** Lag en klasse `Tid` som kan beregne antall sekunder mellom to tidspunkter. Lag klassen så enkel som mulig—det eneste du trenger er en *constructor* og en `operator-`. Lag både klassedeklarasjon og medlemsfunksjoner.

(b) **[8%]** Lag deklarasjonen for en klasse `Telefon` som kan representere et internt apparat. Hvert objekt av denne klassen skal kunne initialiseres (med et internnummer, og til null tellerskritt), det skal kunne akkumulere antall tellerskritt via en dertil egnet medlemsfunksjon, og det skal kunne skrive ut en "faktura" som inneholder internnummer, totalt antall registrerte tellerskritt og total pris. (Tenk spesielt over hvordan du vil løse akkumulering av tellerskritt! Klassen skal selv holde orden på kostnadene ved en samtale gitt start- og sluttidspunkt og oppringt nummer.)

(c) **[8%]** Lag et lite hovedprogram som bruker klassen. Dette programmet skal lese en fil som gitt innledningsvis, registrere samtaler til de forskjellige internnumrene og tilslutt skrive ut "fakturaer" for de enkelte. *Hint:* Lag en tabell av `Telefon`-objekter, eller enda bedre, en tabell av pekere til slike.

(d) **[8%]** Implementer samtlige funksjoner til klassen `Telefon`.

(e) **[6%]** Beskriv hvordan du ville implementere en mer detaljert faktura, f.eks. en der tellerskrittene også blir spesifisert etter type (dvs. intern, riks eller utland).