DATO: 4 desember 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. *Kommenter rikelig!*

Oppgave 1

(a) **[2%]** Beskriv tre måter å overføre et flyttall som en parameter til en C++-funksjon på, både fra kompilatorens synspunkt og fra programmererens. Fordeler og ulemper.

(b) **[2%]** Hva skjer hvis vi prøver å includere for mange `.h`-filer i begynnelsen av programkoden vår?

(c) **[2%]** Hva gjør `sizeof`-operatoren i C++?

(d) **[2%]** Hva er "`this`" i C++?

(e) **[2%]** Skriv kode for å skrive til skjermen flyttallet `Sum` høyrejustert i et felt med bredde 10, og med to desimaler etter komma.

(f) **[2%]** Hva er det i C++ som gjør at vi kan behandle en tekstfil og tastaturet på omtrent samme måte? Forklar kort hvordan dette er ordnet i standard-bibliotekene.

(g) **[2%]** Vi lager en basisklasse `Papir`, og en klasse `Bokside` avledet av `Papir`. Kan vi behandle et objekt av `Papir`-klassen som om det var et `Bokside`-objekt? Kan vi behandle et objekt av `Bokside`-klassen som om det var et `Papir`-objekt? Forklar hvorfor (ikke).

(h) **[2%]** Hva gjør følgende rutine?

```
char *Mystisk(int x)
{
  static char str[33];
  char *p=str+33;
  *--p = 0;
  do {
    *--p = '0'+ x%2;
    x /= 2;
  } while (x);
  return p;
}
```

(i) **[2%]** I følgende deklarasjoner, hva er `V`, `W`, `X`, `Y` og `Z`?

(i) `char* V,W;`

(ii) `float *X[4][3];`

(iii) `int *Y(int&);`

(iv) `int (*Z)(int);`

(j) [2%] Gitt en klasse `class Tall`, skriv fullt kvalifiserte (dvs. ta med klassetilhørighetsoperator) prototyper for operatorer som kan

(i) multiplisere to `Tall`-objekter og returnere et `Tall`,

(ii) multiplisere et `Tall`-objekt og et heltall og returnere et `Tall`,

(iii) multiplisere et heltall og et `Tall`-objekt og returnere et `Tall`,

(iv) multiplisere et `Tall`-objekt til et annet (operatoren er `*=`),

(v) Lese et `Tall` fra en `istream`.

Oppgave 2

(a) [5%] Lag en rutine som konverterer en tekststreng av binære siffer (0 og 1) til et heltall. Rutinen skal ha prototypen

```
int LesBin(char *streng);
```

For eksempel, hvis rutinen kalles opp med strengen `"101010"`, skal det returnerte tallet ha verdien 42. Strengen `"11111"` har verdien 31.

(b) [5%] Lag en innlesningsrutine (fra en `istream`) som kan lese binærtall. Denne skal kunne brukes f. eks. slik:

```
int i;
i = LesBin(cin);
```

Bruk rutinen fra deloppgave (a) til å gjøre dette!

(c) [5%] En mer brukervennlig metode for innlesning av binærtall ville kanskje være å lage en egen *klasse*, slik at man kunne skrive

```
BinTall b;
cin >> b;
int i = b;
```

Hvordan ville du gjøre dette? Det er ikke nødvendig å lage komplett kode, men skriv gjerne hvis det letter fremstillingen.

Oppgave 3

En enkel måte for å sjekke dataintegritet er å lage en *sjekksum*. Sjekksummen av en serie tall er summen av tallene, modulo en sjekksumgrense.

(a) **[10%]** En *fil* kan sees på som en serie av `unsigned char`. Lag en rutine som tar en referanse til en åpen `ifstream` som parameter og returnerer sjekksummen til filen. Sett sjekksumgrensen til 256. (Husk at en `unsigned char` har tallverdi i området $0 \ldots 255$.)

(b) **[5%]** Skriv et lite hovedprogram som bruker rutinen.

(c) **[5%]** Beskriv hva som er nødvendig for å lage en klasse nedarvet fra `ifstream` som fungerer som vanlig, men som også har en medlemsfunksjon for å beregne filens sjekksum. Skriv gjerne kode hvis det letter forklaringen. (*Hint:* constructorer arves ikke!)

Oppgave 4

En ofte brukt matematisk enhet som egner seg godt for C++ er *matriser*. En $M \times N$-matrise består av $M \cdot N$ flyttall. Hvis $\mathbf{A}$ er en slik matrise, vil den bestå av elementene a_{11}, $a_{12} \ldots a_{MN}$, slik:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ & & \vdots & \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{bmatrix}$$

(a) **[5%]** Lag en klassedeklarasjon for matriser. Kall klassen `Matrise`. Dimensjonene M og N bør gis som parametre til en *constructor*. Følgende operatorer og medlemsfunksjoner skal være med i klassen:

- `operator+` og `operator-` for addisjon hhv. subtraksjon av matriser (returverdi: en ny matrise);
- `operator*` for multiplikasjon av to matriser (returverdi: en ny matrise);
- `operator*` for multiplikasjon av en matrise med et flyttall (returverdi: en ny matrise);
- `operator*=` for å multiplisere en matrise inn i en annen (returverdi er her en referanse til den første matrisen);
- `operator*=` for å multiplisere et flyttall inn i en matrise (returverdi er her en referanse til matrisen);
- `operator()(int m,int n)` for henting/setting av enkeltelementer;
- en funksjon `invers` for å invertere en matrise (returverdi: en ny matrise);
- innlesningsoperator;
- utskriftsoperator.

Legg til de nødvendige andre funksjoner du trenger (constructorer, destructor etc.). Vi må i tillegg ha muligheter for å finne størrelsen (M og N) til en eksisterende matrise via dertil egnede funksjoner.

(b) **[5%]** Siden størrelsen av en matrise ikke er bestemt på forhånd, må dynamisk minneallokering benyttes. Lag *constructoren* til klassen `Matrise`. Elementene skal initialiseres til null.

(c) **[5%]** Addisjon av to matriser kan kun gjøres hvis de er av eksakt samme dimensjon (M og N er like for de to matrisene). Hvis matrisene er like store, består addisjonen i å addere ledd for ledd med samme indeks i de to matrisene. Lag funksjonen (operatoren) som adderer to matriser (la programmet stoppe med en feilutskrift dersom man forsøker å addere to matriser av forskjellig størrelse).

(d) **[5%]** Ved multiplikasjon av en matrise med et flyttall, multipliseres flyttallet til alle elementene i matrisen. Lag operatorene `operator*` og `operator*=` for dette.

(e) **[5%]** Lag operatoren som henter/setter matriseelementer. Sjekking av indeksgrenser skal utføres.

(f) **[5%]** Avled en klasse `KvadratiskMatrise` av matriseklassen. Eneste forskjellen er at her skal dimensjonene M og N alltid være like. Skriv kroppen til samtlige funksjoner som er nødvendige for denne klassen. *NB:* Ikke gjør dette vanskeligere enn nødvendig!

(g) **[5%]** En *enhetsmatrise* er en kvadratisk matrise hvor alle elementene langs hoveddiagonalen (dvs. hvor indeksene m og n er like) er lik 1, ellers null. Avled klassen `EnhetsMatrise` av den kvadratiske klassen, og skriv også her samtlige medlemsfunksjoner.

(h) **[5%]** Lag utskriftsoperatoren for matriseklassen, slik at f.eks. denne matrisen:

$$\begin{bmatrix} 1.23 & 2 & 10 \\ 2 & -3.4 & 100 \\ 3.14 & 2.08 & -1000 \end{bmatrix}$$

skrives ut slik:

```
1.23      2.00     10.00
2.00     -3.40    100.00
3.14      2.08  -1000.00
```

(det vil si, hvert tall skrives ut i et felt som er 10 karakterer bredt, og med to desimaler etter komma).

(i) **[5%]** Lag et lite program som genererer matrisen ovenfor, multipliserer denne med sin egen inversmatrise og deretter med 5, legger til en enhetsmatrise av samme størrelse, og tilslutt skriver resultatet til skjermen.