DATO: 22 mai 1995

HØGSKOLEN I STAVANGER
AVDELING FOR TEKNISK-NATURVITENSKAPELIGE FAG
EKSAMEN I: 4495 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.

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

(a) [5%] Beskriv hva slags variable `p` - `v` nedenfor er:

- `int* p,q;`
- `const r = 4;`
- `const int &s = r;`
- `float (*t)[10], *u[10];`
- `void (*v)(int,int);`

(b) [5%] Operatorene "`::`" og "`.`" er begge brukt i forbindelse med klasser i C++. Forklar kort når og hvordan disse brukes.

(c) [5%] Hvis en `int` blir representert med 24 bit på en datamaskin, hva blir gyldig tallområde for en `int` og en `unsigned int`?

(d) [5%] Oversett følgende logiske uttrykk til C++:

- minst én av `j`, `k` og `l` er odde
- `j`, `k` og `l` er alle odde
- `Karakter` er ikke et siffer (dvs. ikke '0'-'9')
- `funnet` er minst det dobbelte av `gitt`, eller `funnet` er null og `gitt` er negativ
- `-grense` ≤ `sum` ≤ `grense`

(e) [5%] Forklar forskjellen på en deklarasjon og en definisjon i C++.

(f) [5%] Gitt en klassedeklarasjon som følger:

```
class Klasse {
  int A, B;
public:
  Klasse(int a=0, int b=0) : A(a+1), B(b) {};
  //.. andre medlemsfunksjoner
};
```

Hva får vi ved følgende linjer i et hovedprogram?

```
  Klasse K[3];
  Klasse H(4);
```

Oppgave 2

Gitt filen `kort.h` som følger:

```
// kort.h
#ifndef _KORT_H
#define _KORT_H
enum Farge {Spar, Hjerter, Ruter, Klover};

struct Kort {
  Farge farge;    // farge er en av Spar, Hjerter, Ruter, Klover,
  int   verdi;    // verdi er mellom 1 og 13
};

ostream& operator<< (ostream&,Kort);
#endif
```

For å implementere en kortstokk med 52 kort skal klassen `Kortstokk` basert på `Kort` lages. Det er 13 kort av hver av de fire fargene (`Spar`, `Hjerter`, `Ruter` og `Klover`), med verdier (`verdi`) lik 1 ... 13 for hhv. ess, to, tre, ..., knekt, dame, konge.

Kortstokken skal kunne initieres via sin constructor, den skal kunne stokkes, og den skal kunne returnere et kort v.hj.a. `operator[]`, med en indeks mellom 1 og 52.

(a) **[10%]** Lag en deklarasjon for klassen `Kortstokk`. Tenk godt igjennom hvordan du vil holde orden på de 52 forskjellige kortene. Kall filen `kortstok.h`.

(b) **[10%]** Skriv implementasjonsfilen (`kortstok.cpp`) for klassen.

Merk: Følgende hovedprogram skal kunne skrive ut kortene i kortstokken:

```
void main()
{
  Kortstokk stokken;

  stokken.stokke();
  for (int i=1; i<=52; i++)
  {
    Kort k = stokken[i];
    cout << k << endl;
  }
}
```

Hint: Stokkefunksjonen kan realiseres via følgende algoritme: "La en teller t gå fra første til siste kort. For hver verdi av t, bytt innholdet av kort t og et tilfeldig kort." Du kan bruke funksjonen `int random(int Max)` fra filen `stdlib.h` som returnerer et tilfeldig tall mellom 0 og `Max-1` for dette.

(c) **[10%]** Implementer nødvendige rutiner i filen `kort.cpp`.

Oppgave 3

NB: Les hele oppgaven før du begynner besvarelsen.

Hendelsesdrevet simulering (event driven simulation) er en meget brukt metode for simulering av virkelige prosesser, som egner seg ypperlig for objektorientert programmering. Kort fortalt består en slik simulering av et antall prosesser som reagerer på stimuli, og en sentral "orkesterleder" (scheduler) som står for sendingen av disse stimuli.

Vi skal i denne oppgaven lage et meget enkelt slikt system for styring av et sentrallåssystem i et bygg. Vi skal kunne styre forskjellige dører og vinduer i bygget ved å sende dem beskjeder om å åpne, lukke eller låse fra sentralen vår. Den sentrale kontrollen kan i dette enkle tilfellet implementeres som en serie linjer i et hovedprogram — et eksempel er gitt i deloppgave **(c)** under.

Grunnlaget for alle dører og vinduer i bygget er en abstrakt klasse `Aapning`, deklarert som følger i filen `aapning.h`.

```
// aapning.h

#ifndef _AAPNING_H
#define _AAPNING_H

enum Beskjed {AAPNE,LUKK,LAAS,LAASOPP};
enum Status {AAPEN,LUKKET,LAAST};

class Aapning {
protected:
  int nummer;
  int etasje;
  Status status;
public:
  Aapning(int nummeret,int etasjen,Status statusen=LUKKET);
  virtual void SkrivStatus(void) const;
  virtual Status Utfoer(Beskjed)=0;
};

#endif
```

Funksjonen `Utfoer()` er ment å skulle utføre gitt operasjon og skrive ut en melding til skjermen; funksjonen skal returnere status for døren/vinduet etter utført operasjon (om det nå er åpent, lukket eller låst). Funksjonen `SkrivStatus()` skriver status til skjermen, og kan med fordel kalles fra `Utfoer()`.

Både dører og vinduer i bygget finnes i to typer: automatiske og manuelle. De manuelle dører/vinduer kan ikke styres av programmet; dette må ordnes ved at vaktmester gis beskjed om hva som skal gjøres. Anta at vaktmesteren kan få beskjeder direkte via `cerr`. Når det gjelder automatiske dører/vinduer, kan for eksemplets skyld dette utføres kun ved at variabelen `status` forandres. Dette vil så bli reflektert når medlemsfunksjonen `SkrivStatus` kalles opp.

Merk: Alle datamedlemmene i klassen `Aapning` er deklarert som `protected` for å lette adkomsten til avledete klasser.

(a) [10%] Lag og tegn et arvehierarki med basis i klassen `Aapning` ovenfor, som kan beskrive klassene under — behold abstraksjonen så langt nedover du kan i hierarkiet. Beskriv eventuelle nye datamedlemmer du deklarerer for de avledete klassene, og spesifiser hvilke funksjoner som må (re-)defineres.

- Dør med automatisk åpne/lukke/låse-kontroll
- Vindu med automatisk åpne/lukke/låse-kontroll
- Manuell dør
- Manuelt vindu

(b) [10%] Skriv klassedeklarasjoner for alle klassene du har spesifisert ovenfor.

(c) [5%] En del av et hovedprogram gis under. Utvid programmet så det blir en demonstrasjon på bruk av klassene du har laget.

```
#include "..." // nødvendige .h-filer

void main(void)
{
  Aapning *aapninger[MAXANTALL];
  // ..
  aapninger[0] = new ... // definerer dører/vinduer
  // ..
  aapninger[0]->Utfoer(LAAS);
  // ..
}
```

(d) [10%] Implementer (skriv) såpass mange medlemsfunksjoner for klassene du har deklarert at det er klart du har en forståelse for løsningsmetoden (du trenger ikke gjenta deg selv).

(e) [5%] Oppgaven blir langt mer interessant hvis også tidsaspektet tas med. Beskriv *kort* hvordan du kunne tenke deg å implementere kontrollprogrammet (styringen) via en eller flere klasser i C++, slik at programmet også kunne styre tidspunkter for de forskjellige begivenhetene. Eksempelvis kan man tenke seg at alle vinduer og dører skal låses klokken 22.00. Tegn gjerne skjemaer, men ikke skriv noe programkode her.