UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

PRØVEEKSAMEN i INF1000
23. november 2004 kl. 14.00 – 17.00

- Dine svar skal skrives på disse oppgavearkene, og ikke på separate ark. Dette gjelder både spørsmål med avkrysningssvar og spørsmål hvor du bes om å skrive programkode. I de oppgavene hvor det skal skrives programkode, anbefales det at du først skriver en kladd på eget ark før du fører svaret inn i disse oppgavearkene på avsatt plass.

- Noen av spørsmålene er flervalgsoppgaver. På disse oppgavene får du poeng etter hvor mange korrekte svar du gir. Du får ikke poeng hvis du lar være å besvare et spørsmål, eller dersom du krysser av begge svaralternativer. Hvis du har satt et kryss i en avkrysningsboks og etterpå finner ut at du ikke ønsket å krysse av der, kan du skrive "FEIL" like til venstre for den aktuelle avkrysningsboksen.

- På denne prøveeksamenen står det maksimum antall poeng du kan få på hver deloppgave hvis alt er riktig. Dette skal du benytte under gjennomgangen på Sophus Lie kl 1800 – 2000 til selv å være sensor for din oppgave.

- Disponer tiden fornuftig på prøveeksamen og eksamen! Løs gjerne oppgavene i den rekkefølgen de står, men sørg for å sette av god tid til de senere oppgavene hvor du bl.a. skal programmere selv. Ikke skriv mer programkode enn nødvendig! Du får ikke tillegg for å legge til ting som det ikke er bedt om. Det er ikke nødvendig å kommentere programkoden annet enn der hvor det er bedt om.

- På eksamen er alle trykte og skrevne hjelpemidler tillatt (det inkluderer f.eks. læreboka, forelesningsnotater og egne notater du måtte ha gjort), så bruk gjerne slike hjelpemidler når du løser denne oppgaven også. Ikke bruk datamaskin når du løser prøveeksamen – det er bare å lure seg selv siden poenget er at denne prøveeksamen skal være så lik en ordentlig eksamen som mulig.

- Pizza serveres kl 1700 – 1800 i kantina i Biologibygningen (på andre siden av bilveien på baksiden av Sophus Lies auditorium). Det er bestilt til kl 17, men fordi det er snakk om mange pizzaer kan vi ikke regne med at alt ankommer helt presis. Vi ber om forståelse for at vi ikke kan tilfredsstille alles ønsker om å få sin favorittpizza. Vi har bestilt noen vegetarpizzaer også. Hvis du ønsker drikke til maten må du bringe selv.

Oppgave 1 (max: 7 poeng)

Er dette lovlige deklarasjoner (anta at de foretas inni en metode)?

JA	NEI	
☐	☐	`int i;` `int k = i;`
☐	☐	`int i, j = 4;` `int k = j;`
☐	☐	`double[] x = new double[1];`
☐	☐	`String[] s = new char[10];`
☐	☐	`String s = {”Arne”, ”Ole”};`
☐	☐	`int i = new int;`
☐	☐	`char[] c = null;`

Oppgave 2 (max: 15 poeng)

Hvor mange ganger blir ”INF1000” skrevet ut?

a)
```
for (int i=0; i<4; ++i) {
   System.out.println(”INF1000”);
}
```

Svar: .....

b)
```
for (int i=1; i<4; i++) {
   for (int j=1; j<5; j++) {
      System.out.println(”INF1000”);
   }
}
```

Svar: .....

c)
```
for (int i=0; i<4; i+=2) {
   for (int j=0; i+j<4; j+=i+1) {
      System.out.println(”INF1000”);
   }
}
```

Svar: .....

Oppgave 3 (max: 5 poeng)

Anta at følgende kodelinjer utføres:

```
int x = 22;
int y = x / 27;
int z = y * 27;
boolean test = (x == z || y > 0);
```

Hva er verdien til variabelen **`test`** rett etterpå?

Svar: .....

Oppgave 4 (max: 10 poeng)

Anta at følgende setninger utføres:

```
int k = 33;
int i = 2*k;
int j = k++;
if (k>j) j += 67;
k = k + j + i++;
int verdi = ++k;
```

Hva er verdien til variabelen **`verdi`** like etter at setningene over er utført?

Svar: .....

Oppgave 5 (max: 16 poeng)

Anta at følgende er deklarert i en metode i et program:

```
int[] vekt = new int[100];
int sum = 0;
```

Anta at arrayen **vekt** fylles opp med tilfeldige verdier. Vil alternativene under alltid gi som resultat at **sum** inneholder summen av alle elementene i arrayen **vekt**?

JA **NEI**

☐ ☐
```
int k = 0;
while (k < vekt.length) {
   sum = sum + vekt[k];
}
```

☐ ☐
```
for (int k = vekt.length; k > 0; k--) {
   sum += vekt[k-1];
}
```

☐ ☐
```
int k = 0;
do {
   k++;
   sum = sum + vekt[k];
} while (k < vekt.length);
```

☐ ☐
```
int k = 0;
do {
    sum = sum + vekt[k];
    k++;
} while (k < vekt.length);
```

☐ ☐
```
for (k = 0; k < vekt.length; k++) {
   if (vekt[k] > 0) {
       sum += vekt[k];
   } else {
      sum -= vekt[k];
   }
}
```

☐ ☐
```
int k = 0;
for (int i = 0; i < vekt.length; i++) {
   sum = vekt[k++];
}
```

☐ ☐
```
for (int i = 0; i < vekt.length - 1; i += 2) {
   sum += vekt[i] + vekt[i+1];
}
```

☐ ☐
```
boolean ferdig = false;
int k = 0;
while (!ferdig) {
   if (k > 99) {
      ferdig = true;
   } else {
      sum = sum + vekt[k++];
   }
}
```

Oppgave 6 (max: 15 poeng)

Vi kan lage en sekvens med heltall på følgende måte: la de to første elementene i sekvensen begge være 1 og la deretter hvert element være summen av de to foregående. Den sekvensen vi da får starter slik: 1, 1, 2, 3, 5, 8, 13, ... og kalles Fibonacci-sekvensen. Du skal nå lage en metode som beregner de n første elementene i Fibonacci-sekvensen og returnerer disse i en heltallsarray av lengde n+1 (det første elementet i arrayen, svarende til posisjon 0, skal ikke benyttes). Posisjon k i arrayen som returneres skal inneholde det k'te elementet i Fibonacci-sekvensen for k=1,....,n. Du kan anta at $n \geq 2$.

Svar:

```
int[] fibonacci (int n) {




}
```

Oppgave 7 (max: 15 poeng)

Skriv ferdig metoden under, som teller opp hvor mange ganger tegnet i char-variabelen **`tegn`** forekommer i tekststrengen **`tekst`**, og returnerer dette antallet. For eksempel skal kallet **`antallForekomster ("Engelske", 'e')`** returnere verdien 2. Hvis **`tekst == null`** så skal metoden returnere verdien 0.

Svar:

```
int antallForekomster (String tekst, char tegn) {









}
```

Oppgave 8 (max: 15 poeng)

Norge består av 19 fylker som hver består av et antall kommuner. Anta at vi modellerer dette ved hjelp av tre klasser **`Land`**, **`Fylke`** og **`Kommune`** slik at Norge representeres ved et objekt av klassen **`Land`**, hvert fylke er representert ved et objekt av klassen **`Fylke`** og hver kommune er representert ved et objekt av klassen **`Kommune`**. Lag et UML klassediagram som viser forholdet mellom de tre klassene **`Land`**, **`Fylke`** og **`Kommune`**, og få på riktige antall i hver ende av forholdene.

Svar:

Oppgave 9 (max: 15 poeng)

Anta at følgende program utføres:

```
class StudentSystem {
    public static void main (String[] args) {
       Studieprogram inf = new Studieprogram("Informatikk");
       Studieprogram bio = new Studieprogram("Biologi");

       new Student("Kari Olsen", inf);
       new Student("Eva Larsen", inf);
       new Student("Per Jensen", bio);

       bio.liste();
       inf.liste();
    }
}

class Studieprogram {
   String tittel;
   Student[] medlemmer;
   int antall;

   Studieprogram (String tittel) {
      this.tittel = tittel;
      medlemmer = new Student[100];
      antall = 0;
   }

   void meldInnStudent(Student stud) {
      medlemmer[antall] = stud;
      antall++;
   }

   void liste() {
      for (int i=0; i<antall; i++) {
         System.out.println(medlemmer[i].somTekst());
      }
   }
}

class Student {
   String navn;

   Student (String navn, Studieprogram sp) {
      this.navn = navn;
      sp.meldInnStudent(this);
   }

   String somTekst() {
      return navn;
   }
}
```

Skriver programmet noe ut på skjermen, og isåfall hva?
Svar:

Oppgave 10 (max: 15 poeng)

Anta at du har deklarert en HashMap:

```
HashMap cdSamling = new HashMap();
```

og at du legger inn informasjon om CD'ene dine i HashMapen, med platens tittel som nøkkel og artistnavnet som verdi (du antar at alle platene dine alle har ulik tittel). Et eksempel er

```
cdSamling.put("Not going under", "Maria Arredondo");
```

Skriv noen programsetninger som:

- først ber om og leser inn et artistnavn fra tastatur (f.eks. "Maria Arredondo").
- deretter går gjennom HashMapen og skriver ut titlene til alle platene du har registrert med denne artisten. Hint: husk at du vil ha tak i både nøklene og verdiene når du løper gjennom HashMapen. Det kan derfor lønne seg å lage en Iterator over nøklene (platetitlene).

Du kan anta at programsetningene plasseres slik i programmet at de har tilgang til HashMapen **`cdSamling`**.

Svar:

Oppgave 11 (max: 15 poeng)

I programmet nedenfor skal du lage en konstruktør til klassen Student. Konstruktøren skal ha studentens navn som parameter og skal initiere objektvariabelen **navn**. Du skal også skrive en objektmetode **void økPoeng(int poeng)** i klassen Student som øker antall studiepoeng med parameterens verdi. I klassen StudentTest skal du så sørge for å få initiere arrayen **uioStud** med 32000 Student-objekter. Studentobjektene i arrayen **uioStud** skal ha hvert sitt studentnavn **Stud-1**, **Stud-2**, **Stud-3**, ... **Stud-32000**. Dermed skal f.eks. **uioStud[252]** peke på et Student-objekt hvor studentens navn er satt lik **Stud-253**. Til sist skal du sette antall studiepoeng til 30 for hver av studentene **Stud-1**, ..., **Stud-25000**.

```
class Student {
   String navn;
   int antallStudiepoeng = 0;

   // Her skal du skrive konstruktøren









   // Her skal du skrive objektmetoden økPoeng









}


class StudentTest {
  Student[] uioStud = new Student[50000];

  public static void main (String[] args) {
    // Her skal du opprette 32000 Student-objekter med navn Stud-1,
    // Stud-2,...,Stud-32000 og legge dem inn i uioStud-arrayen









    // Her skal du øke antall studiepoeng med 30 for studentene
    // med studentnavn Stud-1, Stud-2, ...., Stud-25000.









  }// end main
 }
```

Oppgave 12 (max: 20 poeng)

Filen **`SpamOrd.txt`** inneholder endel *spam-ord* (atskilt av blanke tegn), dvs ord som du ikke forventer å finne i seriøse eposter til deg, men som ofte forekommer i useriøse eposter (såkalt spam-epost). Du skal lage et fullstendig program som leser **`SpamOrd.txt`** og som deretter leser filen **`Epost.txt`** og sjekker om denne inneholder noen av spam-ordene. Programmet skriver til slutt ut på skjermen om filen **`Epost.txt`** inneholdt noen spam-ord eller ikke (du trenger ikke å skrive ut hvilke eller hvor mange spam-ord filen inneholdt). Du kan anta at **`SpamOrd.txt`** maksimalt inneholder 200 ord. I denne oppgaven kan du godt legge all programkoden inn i main-metoden.

Svar:

(forts. neste side)

Oppgave 13 (vanskelig; max 20 poeng)

Skriv metoden under som finner og skriver ut på skjerm hvor mange ulike verdier du har i en array og returnerer dette antallet. Hint: definer en heltallsvariabel **`antUlike`** og initier denne til 0. Løp deretter gjennom alle elementene i arrayen. For hvert verdi du ser øker du **`antUlike`** med 1 dersom det er siste gang denne verdien forekommer i arrayen, dvs hvis ingen elementer med høyere indeks er lik denne verdien (du kan sjekke dette med en ny løkke inni den ytre løkken). Dermed er du sikret at hver verdi bare telles en enkelt gang. Du kan anta at lengden av arrayen **`a`** er minst 1.

Svar:

```
void finnAntallUlikeTall (int[] a) {









 }
```

Oppgave 14 (vanskelig; max 20 poeng)

Et primtall er et heltall $k \geq 2$ som bare er delelig med 1 og seg selv. F.eks. er 5 et primtall fordi det ikke er delelig med noen av tallene 2, 3, 4 (med andre ord: vi kan ikke skrive 5 som et helt multiplum av noen av disse tallene), mens 4 ikke er et primtall fordi det er delelig med 2.

Det minste primtallet er 2. Hvis $k \geq 3$ er et heltall, så er k et primtall hvis det ikke er delelig med noen av tallene 2, 3, ..., k-1. For å sjekke om et tall er delelig med et annet i Java kan vi bruke operatoren **`%`**. Hvis **`a`** og **`b`** er to heltall, så vil **`a % b == 0`** hvis og bare hvis **`a`** er delelig med **`b`**.

Skriv ferdig metoden nedenfor som skriver ut på skjermen de **`n`** første primtallene. F.eks. skal kallet **`skrivPrimtall(5)`** skrive ut tallene 2, 3, 5, 7 og 11. Du kan anta at $n \geq 1$.

Svar:

```
void skrivPrimtall (int n) {









}
```