4. september, 2003

MAT-INF 1100: Obligatorisk oppgave 1

Innleveringsfrist: 12/9-2003, kl. 14:30

Informasjon

Den skriftlige besvarelsen skal leveres på ekspedisjonskontoret i 7. etg. i Niels Henrik Abels hus senest *kl. 14.30 fredag 12/9*. Javaprogrammer og kjøring av slike leveres som utskrift fra skriver, men for hver oppgave skal du også levere en beskrivelse som kommenterer resultatene. Denne *skal* være håndskrevet av deg selv (med mindre du har fått spesiell tillatelse til å levere besvarelsen i en annen form).

Studenter som blir syke eller av andre grunner ønsker å søke om utsettelse eller fritak fra denne obligatoriske oppgaven, må ta kontakt med Heidi Raude (rom B718 Niels Henrik Abels hus, telefon 22 85 59 01, e-post: heidimr@math.uio.no) i god tid før innleveringsfristen.

Det oppfordres til samarbeid underveis i arbeidet med oppgavene, og gruppelærerne har anledning til å svare på generelle spørsmål, men kan ikke servere ferdige løsninger. *Den endelige besvarelsen som du leverer skal utarbeides av deg selv, og du må kunne redegjøre for innholdet ved en eventuell muntlig høring.*

Husk at de to obligatoriske oppgavene i MAT-INF 1100 må begge bestås for å kunne gå opp til endelig eksamen i kurset. *For å få bestått på denne første obligatoriske oppgaven må du gjøre seriøse løsningsforsøk på alle oppgavene, og programmene for minst to av de fem oppgavene bør gi riktig svar.*

Oppgave 1. I matematikk brukes ofte en notasjon for produkter tilsvarende den for summer. Uttrykket $\prod_{i=1}^{6} i$ er definert ved

$$\prod_{i=1}^{6} i = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6.$$

Symbolet $\prod$ er altså helt analogt med $\sum$ bortsett fra at plusstegnene blir endret til multiplikasjonstegn, slik at de aktuelle tallene multipliseres i steden for å adderes. Programmer og skriv ut resultatet av følgende produkter og kommenter resultatene.

a) $\prod_{i=1}^{30} i$.

b) $\prod_{i=3}^{20} 2$.

c) $\prod_{i=1}^{10} i^3$.

d) $\prod_{i=5}^{100} 1/i$.

e) $\prod_{i=-4}^{4} i/(i+1)$.

Oppgave 2. Ut fra teksten i kompendiet er det klart at på en datamaskin vil addisjonen $1 + \epsilon$ bli regnet ut til 1 hvis bare ϵ er liten nok. Skriv et program som kan hjelpe deg til å finne det minste heltallet n slik at $1 + 2^{-n}$ blir 1. Gjør dette både for 32- og 64-bits flyttall (`float`- og `double`-variable i Java). Er svarene rimelige?

Oppgave 3. Binomialkoeffisientene $\binom{n}{i}$ er definert ved

$$\binom{n}{i} = \frac{n!}{i!\,(n-i)!} \tag{1}$$

der $n \geq 0$ er et heltall og i er heltall i intervallet $0 \leq i \leq n$. Binomialkoeffisientene forekommer i mange formler og må derfor derfor ofte beregnes på datamaskin, og siden alle binomialkoeffisienter er heltallige (divisjonen i (1) må altså alltid gå opp) er det rimelig å bruke heltallsvariable til slike beregninger. For små verdier av n og i går dette greit, men for litt større verdier får vi fort problemer fordi teller og nevner i (1) da kan bli større enn de største heltall som kan representeres på maskinen selv om binomialkoeffisienten

i seg selv ikke er så stor. Ved å bruke flyttallsvariable i steden kan vi håndtere litt større tall, men igjen vil vi kunne få for store tall underveis selv om sluttresultatet ikke er for stort. I tillegg vil få avrundingsfeil når vi bruker flyttall.

Det som er uheldig med formelen (1) er at selv om binomialkoeffisienten vi er ute etter ikke nødvendigvis er så stor vil teller og nevner allikevel kunne være svært store. Slikt er generelt uheldig for numeriske beregninger og bør unngås så sant det er mulig og ikke koster for mye i form av ekstra regnetid. Hvis vi ser litt nøyere på formelen (1) så ser vi at vi kan forkorte slik at

$$\binom{n}{i} = \frac{1 \cdot 2 \cdots i \cdot (i+1) \cdots n}{1 \cdot 2 \cdots i \cdot 1 \cdot 2 \cdots (n-i)} = \frac{i+1}{1} \cdot \frac{i+2}{2} \cdots \frac{n}{n-i}.$$

Med andre ord kan vi hjelp av produktnotasjon skrive $\binom{n}{i}$ som

$$\binom{n}{i} = \prod_{j=1}^{n-i} \frac{i+j}{j}.$$

a) Programmer en metode for å beregne binomialkoeffisienter basert på denne formelen, og test metoden din på binomialkoeffisientene

$$\binom{10000}{3} = 166616670000,$$

$$\binom{100000}{80} = 1.353770149276343 \cdot 10^{281},$$

$$\binom{1000}{500} = 2.702882409454366 \cdot 10^{299}.$$

Hvorfor må du bruke flyttall og hvilke resultater får du?

b) Er det nå mulig å få for store tall underveis i beregningene dersom binomialkoeffisienten vi skal beregne ikke er større enn det største flyttallet vi kan representere.

c) Når vi utledet metoden vår forkortet vi $i!$ mot $n!$ i (1) og forenklet på den måten uttrykket for $\binom{n}{i}$. Vi kan utlede en annen metode ved i steden å forkorte $(n-i)!$ mot $n!$. Utled denne metoden på samme måte som over, og diskuter når de to

metodene bør brukes (du trenger ikke programmere den andre metoden, bruk matematiske resonnementer).

Oppgave 4. I denne oppgaven skal vi se hvordan overflow og verdien `NaN` oppfører seg i Java. Hver deloppgave skal programmeres og du skal beskrive resultatet som programmet ditt gir og om dette er rimelig.

a) Bruk variable av typen `long` og multipliser sammen de to tallene 2^{50} og 2^{30}. Sjekk om svaret stemmer med den riktige verdien som er $2^{80} = 1208925819614629174706176$.

b) Bruk `double`-variable, multipliser sammen de to tallene 10^{200} og 10^{300} og skriv ut svaret. Blir det 10^{500}?

c) Adder 1 og ta kvadratroten til svaret i oppgave (b). Hva blir resultatet?

d) Divider 1 med svaret maskinen ga i (c) og skriv ut svaret. Kommentarer?

e) Multipliser svaret fra (c) med 0, skriv ut svaret og kommenter.

f) Multipliser svaret fra (e) med 2 og skriv ut svaret. Kommentarer?

g) Forsøk å utføre de to udefinerte divisjonene $1.0/0.0$ og $1/0$, lagre resultatet i henholdsvis en `double`-variabel og en `long`-variabel og skriv ut resulatet. Kommentarer?

Oppgave 5. En grunnleggende egenskap ved reelle tall er den assosiative loven

$$(x + y) + z = x + (y + z) \tag{2}$$

som sier at det ikke spiller noen rolle hvilken rekkefølge vi utfører addisjoner i. Du skal i denne oppgaven sjekke om denne loven gjelder for flyttall. For å gjøre dette skal du skrive et program som går i en løkke 1000 ganger og hver gang trekker tre tilfeldige tall av typen `double` og så sjekker om loven gjelder (om de to sidene i (2) er like) for disse tallene. Tell opp hvor mange ganger loven svikter, og skriv til slutt ut i hvor mange prosent av tilfellene den sviktet. Skriv også ut ett sett med tre tall der loven sviktet.

Tips: For å trekke tilfeldige tall kan du skrive

```
import java.util.Random
```

aller først på programfila og deklarere en variabel som du kan kalle `tilfeldig` ved

```
Random tilfeldig = new Random();
```

Hvis `x` er definert som en variabel av typen `double`, kan du trekke et tilfeldig tall av type `double` ved å si

```
x = tilfeldig.nextDouble();
```

Et fullstendig program som skriver ut ett tilfeldig tall kan da se ut som

```
import java.util.Random;

class testrandom {

    public static void main (String [ ] args) {
         double x;
         Random tilfeldig = new Random();
         x = tilfeldig.nextDouble();
         System.out.println("Et tilfeldig tall: " + x);
    }
}
```

Lykke til!!