September 8, 2006

MAT-INF 1100: Compulsory assignment 1

Deadline: September 22, 2006, 14:30

Information

The written answers are to be handed in at the office of the Maths Department on the 6th floor (7. etasje in Norwegian) by 14:30 on Friday, September 22. For Javaprograms and output from programs, printouts should be handed in, but for every problem you should also hand in a description with comments on your results. This should be written by yourself (by hand or computer).

Students who become ill or for other reasons wish to apply for postponement or exemption from this assignment should contact Elisabeth Seland (room B726 in Niels Henrik Abels building, telephone 22 85 59 07, email: elisabh@math.uio.no) in good time before the deadline.

Students are encouraged to work together on this assignment, and the teaching assistants will answer general questions, but cannot provide complete solutions. *The final answers that you hand in must be produced by yourself, and you must be able to explain the contents of your answers if you are called in for an oral examination (may happen if there is suspicion of copying).*

Remember that both of the two compulsory assignments in MAT-INF 1100 must be passed before you are allowed to sit the final course exam. *To pass this first assignment you must make serious attempts at solving all the problems, and the programs for at least two of the five problems should produce the right answers.*

Printout of program execution. One part of your answers should consist of printouts of execution of your programs. Such printouts can be produced by the linux-command `script`. If for example you give the command `script printout` what you do thereafter will be recorded in the file `printout`. You quit this mode by typing `exit`. For more info, give the command `man script`.

Problems

Problem 1. In this problem we are going to see how overflow and the value `NaN` behave in Java. In each subproblem you should describe the behaviour of your program and whether this is reasonable.

a) Use variables of type `long` and multiply the two numbers 3^{31} and 3^{11}. Check if the answer agrees with the correct value which is $3^{42} = 109418989131512359209$.

b) Using `double`-variables, multiply the two numbers 10^{300} and 10^{-300} (a number like 10^{300} can be given as a `double` by `1.0e300`) and print the result. Is the answer correct?

c) Multiply the answer in the previous subproblem by itself. What is the result?

d) Divide 10 by the answer produced by the machine in (c) and print the answer. Comments?

e) Multiply the answer from (c) by 0 and print the answer. Comments?

f) Multiply the answer from (e) by itself, print the answer. Comments.

g) Try to perform the two undefined divisions 2/0 and 2.0/0.0, store the result in a `long`-variable and a `double`-variable respectively, and print the result. Comment on the result.

Problem 2. A notation for products similar to that for sums is often used in Mathematics. The expression $\prod_{i=1}^{6} i$ is defined as

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

In other words, the symbol $\prod$ is completely analogous to $\sum$, except that the plus signs are changed to multiplication signs, so that the numbers in question are multiplied, and not added. Write a program that prints out the results of the following products and comment on the results.

a) $\prod_{i=1}^{20} i$.

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

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

d) $\prod_{i=5}^{60} 1/i^5$.

e) $\prod_{i=0}^{10} i/(i-4)$.

Problem 3. The binomial coefficients $\binom{n}{i}$ are defined as

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

where $n \geq 0$ is an integer and i is an integer in the interval $0 \leq i \leq n$. The binomial coefficients turn up in a number of formulae and must therefore often be computed on a computer. Since all binomial coefficients are integers (this means that the division in (1) can never give a remainder), it is reasonable to use integer variables in such computations. For small values of n and i this works well, but for larger values we quickly run into problems because the numerator and denominator in (1) may become larger that the largest integer that can be represented on the computer, even if the binomial coefficient itself may be relatively small. By using floating point numbers we may be able to handle larger numbers, but again we may encounter too big numbers during the computations even if the final result is not so big. In addition, floating point numbers introduce round-off errors.

An unfortunate thing about the formula (1) is that even if the binomial coefficient is small, the numerator and denominator may both be large. In general, this is bad for numerical computations and should be avoided if possible. If we consider the formula (1) in some more detail, we notice that many of the numbers cancel out,

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

Employing the product notation we can therefore write $\binom{n}{i}$ as

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

a) Write a program for computing binomial coefficients based on this formula, and test your method on the coefficients

$$\binom{9998}{4} = 416083629102505,$$
$$\binom{100000}{70} = 8.14900007813826 \cdot 10^{249},$$
$$\binom{1000}{500} = 2.702882409454366 \cdot 10^{299}.$$

Why do you have to use floating point numbers and what results do you get?

b) Is it possible to encounter too large numbers during those computations if the binomial coefficient to be computed is smaller than the largest floating point number that can be represented in the computer?

c) In our derivation we cancelled $i!$ against $n!$ in (1), and thereby obtained the alternative expression for $\binom{n}{i}$. Another method can be derived by cancelling $(n-i)!$ against $n!$ instead. Derive this alternative method in the same way as above, and discuss when the two methods should be used (you don't need to program the other method; argue mathematically).

Problem 4. From the text in the lecture notes it is clear that a computer will obtain the result 3 for the floating point addition $3+\epsilon$ if ϵ is small enough. Write a program that can help you determine the smallest integer n such that $3+2^{-n}$ is computed as 3. Do this both for 32 and 64 bit floating point numbers (`float`- and `double`-variables in Java). Are the answers reasonable?

Problem 5. A fundamental property of real numbers is given by the distributive law

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

In this problem you are going to check whether floating point numbers obey this law. To do this you are going to write a program that runs through a loop 10000 times and each time draws three random numbers of type `double` and then checks whether the law holds (whether the two sides of (2) are equal) for these numbers. Count how many times the law fails, and at the end, print the percentage of times that it failed. Print also a set of three numbers for which the law failed.

Hint: To draw random numbers you can write

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

at the beginning of the program file and declare a variable which you may call `random` by

```
Random random = new Random();
```

If `x` is defined as a variable of type `double`, you can then draw a random number by saying

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

A complete program that prints one random number may look as follows:

```
import java.util.Random;

class testrandom {

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

Good luck!!