ESC101 : Fundamental of Computing

Lab Test for 11th November 2008

Instructions:

1. The duration of the test is 3 hrs (from **2:00 pm to 5:00 pm**).
2. **Directory Structure:** Create a directory in the home and name it with your roll number. For example, if your roll number is $Y8001$, the directory should be named $Y8001$ (`Y` should be upper case). Inside this directory, create three directories named **One**, **Two** and **Three** corresponding to the programs for the three problems of this test. In directory **One**, create file *Taylor.java* for the java program for first problem. In directory **Two**, create file *Fraction.java* for the second problem. In directory **Three**, create file **Routes.java** for the java program of the third problem.
3. Use of good coding practices (indentation, use of methods, and proper naming of variables and methods) carries weightage.

1. **Taylor series expansion of e^x :** (marks = 10)

   In mathematics, the Taylor series is a representation of a function as an infinite sum of terms calculated from the values of its derivatives at a single point (To solve the following problem you need not know the details about the Taylor's series expansion). By using Taylor's series expansion of the function $f(x) = e^x$, we can express e^x as an infinite series summation as follows:

   $e^x = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \ldots$

   Where $n! = n \times (n-1)\times)(n-2) \times \ldots \times 1$.

   It is to be noted here that the accuracy of the calculated value is dependent on the number of terms of the right hand side that are considered.

   Write a JAVA program that takes inputs x and n from the command line and then calculates e^x by adding the first n terms of the series expansion. For example, if the inputs are 2 and 3 then your program should consider only 3 terms of the series as:

   $e^2 = \frac{2^0}{0!} + \frac{2^1}{1!} + \frac{2^2}{2!}$.

   And similarly, if the inputs are 0.5 and 5 then

   $e^{0.5} = \frac{0.5^0}{0!} + \frac{0.5^1}{1!} + \frac{0.5^2}{2!} + \frac{0.5^3}{3!} + \frac{0.5^4}{4!}$

2. **Positive Fraction :** (marks = 12)

   Define a *class* called *Fraction* which defines a rational numbers . A fraction has two integer attributes namely *numerator* and *denominator* (you need not consider the sign of the fraction here because we are considering only positive fraction). The class should have two constructors. One constructor with two parameters to initialize the numerator and denominator respectively. And the other constructor should take one parameter and initialize the numerator with the passed value and the denominator with 1.

   You should design and implement the following *non-static* methods:

   (a) **display**: Displays the fraction. For example, if the numerator is 3 and denominator is 5 then the output should be 3/5.

   (b) **add**: which adds a given fraction to *this* fraction.

   (c) **multiply**: which multiplies *this* fraction by a given fraction.

   (d) **isEqual**: Used to check whether *this* fraction is equal to a given fraction.

The main method should take four positive integers `a,b,c,d` from command line, and create two fractions : first fraction `a/b` and second fraction `c/d`. The program should perform the following steps in order.

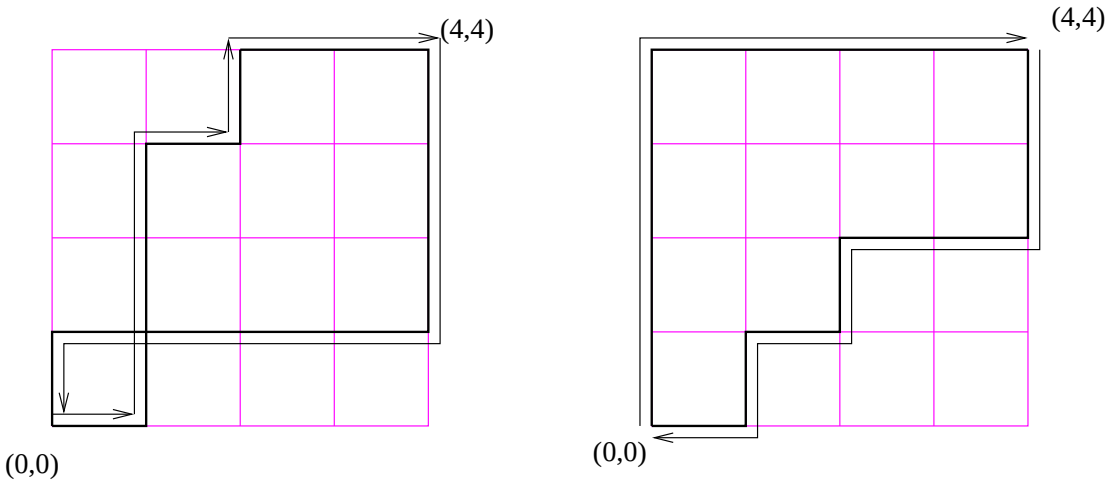
- The program should determine of first fraction is same as the second fraction.
- The program should add second fraction to first fraction, and display the first fraction after the operation.
- It should then multiply the first fraction by the second, and display the first fraction after the operation.

Note : The fractions are not necessarily to be in reduced form. It is perfectly fine even if you do not calculate LCM while adding the two fractions. And similarly, you may avoid calculating GCD while testing for equality. **NO** marks will be deducted for the same.

3. **Counting the number of shortest routes** (marks=18)

Given a $m \times n$ grid, a mouse is standing at the origin, and there is a piece of cheese at the top right corner (m, n). The mouse has to get the piece of cheese and then return to origin. The mouse follows a shortest route during its journey from origin to cheese and back.

For example, the following figure gives two such shortest routes in a (4×4) grid.



Write a program which reads the values of m, n from command line and prints the number of shortest routes for the journey of mouse : origin $\rightarrow$ cheese $\rightarrow$ origin.