Middle East Technical University
Department of Computer Engineering

CENG 140

Spring '10

Take Home Exam 1

REGULATIONS

Due date: 19 April 2010, Monday
(Not subject to postpone)

Submission: Electronically. You will be submitting your program source code written in a file which you will name as `the1.c` through the cow web system. Resubmission is allowed (till the last moment of the due date), The last will replace the previous.

Team: There is **no** teaming up. The take home exam has to be done/turned in individually.

Cheating: This is an exam: all parts involved (source(s) and receiver(s)) get zero+parts may be subject to disciplinary action.

PROBLEM

Quite often we buy stuff by installments. The idea is that somebody is lending you money, after a quanta of time your debt is increased by an additional amount of money which is linearly calculated based on the amount you borrowed. You pay a certain amount of money (the *installment*). Your debt (which was ⟨the money you borrowed⟩ + ⟨interest for that time quanta⟩) now is decreased by an amount of installment. The time quanta is usually a month. For simplicity let us denote the borrowed amount (the *loan)* with L, the montly interest rate r, the monthly down payment p.

So, lets tabulate what is going on:

Time instance	**Debt now**	**You pay now**	**Debt remaining**
Contract start	L	0	L
After 1. month	$L + r \times L$	p	$L + r \times L - p$
After 2. month	$(L + r \times L - p) + r \times (L + r \times L - p)$	p	$(L + r \times L - p) + r \times (L + r \times L - p) - p$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

This gets cluttered quite fast so, let us define $\xi = (1 + r)$ and drop the $\times$ signs. This simplifies the outlook and we have:

Time instance	**Debt now**	**You pay now**	**Debt remaining**
Contract start	L	0	L
After 1. month	ξL	p	$\xi L - p$
After 2. month	$\xi(\xi L - p)$	p	$\xi(\xi L - p) - p$
After 3. month	$\xi(\xi(\xi L - p) - p)$	p	$\xi(\xi(\xi L - p) - p) - p$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

You easily discover that after n. month the debt remaining is:

$$\xi^n L - \xi^{n-1} p - \xi^{n-2} p - \cdots - \xi p - p$$

Which can be rewritten as:

$$\xi^n L - (\xi^{n-1} + \xi^{n-2} + \cdots + \xi + 1)p$$

The sum in the parenthesis is a geometric sum and is easily calculable:

$$(\xi^{n-1} + \xi^{n-2} + \cdots + \xi + 1) = \frac{\xi^n - 1}{\xi - 1}$$

So we obtain the debt after $n.$ month as:

$$\xi^n L - \frac{\xi^n - 1}{\xi - 1}p$$

Now the idea is to have this debt amount zeroed out after the $n.$ month payment. So we have a closed function:

$$\xi^n L - \frac{\xi^n - 1}{\xi - 1}p = 0$$

The function is in four variables, namely L, n, ξ and p. So, given any three of them it should be possible to determine the fourth (if exists).

Having L or p as unknown and solving the equation for any of them is extremely simple. Having n as unknown and solving for it is simple too, since it appears only as ξ^n. The fourth option though is not so simple. Various approximated solution techniques can be thought of. What we ask you to do is to do a Newton-Raphson method root finding for this case. Extensive reference material exists on the web. Here are a few (but good):

- `http://en.wikipedia.org/wiki/Newton's_method`
- `http://numericalmethods.eng.usf.edu/topics/newton_raphson.html`
- `http://www.statemaster.com/encyclopedia/Newton's-method`

SPECIFICATIONS

- You will be given four numbers in input. The first is an integer in the range [1-4], the other three are floating points. You shall treat them as `doubles`. Assume an order of L, n, p, r. The first number is denoting which is missing. The remaining three floats are those values, in the above order, skipping the missing one. So, for example if the input is
  `3 100000.0 360 0.00583`
  that means the Loan is 100000.0, the payment will last 360 months (30 years), the monthly interest rate is 0.00583 (just for your information, that is 7% annually). You are expected to determine p, namely the amount of monthly payment.

- The output that you will generate is the missing value. For the example above that is p, and after you do the necessary calculation you are expected to output the value of `665.0` (for this particular example).

- Your program will be tested with data that provides exactly one solution. No obscure data will be given for testing (e.g. negative interest rate or down payment, etc). The n value will be given as an integer represented as a float, and if the test data requires you to determine n the result will always be an integer. Though you are allowed to output it in floating point notation.

- For a particular test data, the test data and your output (the missing value) will altogether be put back into the equation and the outcome will be compared to 0.0 . Here, due to errors in the calculations we will tolerate errors $< 10^{-5}$

- To keep precision at maximum, you are advised to choose `double` for all type of floating point variables and calculations.