Problem Set 1 **Due: Feb 1** **cs2me3 Winter 2006**

Instructions

1. You are encouraged to work in groups of two. If you cannot find a partner, you can work alone.
2. Please submit one copy of the assignment; if you are working with a partner, both names should appear on the assignment.
3. For **Part A** of the assignment, you must submit an electronic copy of your Java application via WebCT (by the time of the lecture on the due date of the assignment).

Part A

Write a Java application which multiplies polynomials.

Your program should work as follows: given an ASCII text file as input, `polynomials.txt` containing the pairs of polynomials to be multiplied, it outputs the results to the standard output.

The polynomials should be given as vectors of coefficients; for example, $x^5 + 3x^2 - 2x + 89$ should be represented as $(1, 0, 0, 3, -2, 89)$. The input file (`polynomials.txt`) should be presented as follows:

```
(2,1)(-2,3,3,2)
(1,2,5,4,3)(1)
(2,3)(2,0,0)
END
```

and the output (to the standard output) should be:

```
(-4,4,9,7,2)
(1,2,5,4,3)
(4,6,0,0)
```

Part B

1. Invariants.

   (a) Suppose that the positive integer n is odd. Write the numbers $1, 2, \ldots, 2n$ on the blackboard. Then pick any two numbers a, b, erase them, and write $|a-b|$ instead. Continue repeating this until just one number remains on the blackboard; show that this number is odd.

   Solution: Consider the following simple loop invariant: let S be the sum of the numbers currently on the blackboard; then "S is odd" is a loop invariant of the procedure. Now we prove that this loop invariant holds. Basis Case: $S = 1 + 2 + \cdots + 2n = n(2n+1)$ which is odd. Induction Step: assume S is odd, let S' be the result of one more iteration, so $S' = S + |a-b| - a - b = S - 2\min(a, b)$, and since $2\min(a, b)$ is even, and S was odd by the induction hypothesis, it follows that S' must be odd as well. At the end, when there is just one number left, say x, $S = x$, so x is odd.

   (b) In the Parliament of some country, each member has at most three enemies. Show that the house can be separated into two houses, so that each member has at most one enemy in his own house.

   Solution: To solve this problem we must provide both an algorithm and an invariant for it. The algorithm works as follows: initially divide the parlament into any two groups. Let H be the total sum of enemies that each member has in his own house. Now repeat the following loop: while there is an m which has at least two enemies in his own house, move m to the other house (where m must have at most one enemy). Thus, when m switches houses, H decreases. Here the invariant is "H decreases monotonically." Now we know that a sequence of positive integers cannot decrease for ever, so when H reaches its absolute minimum, we obtain the required distribution.

2. Induction.

   (a) The *Fibonacci* sequence is defined as follows: $f_0 = 0$ and $f_1 = 1$ and $f_{i+2} = f_{i+1} + f_i$, $i \geq 0$. Prove the following:

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n = \begin{pmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{pmatrix}.$$

   Solution: The Basis Case is $n = 1$, and it is immediate. For the induction step, assume the equality holds for exponent n, and show that it holds for exponent $n+1$:

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n+1} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} f_{n+1} + f_n & f_{n+1} \\ f_n + f_{n-1} & f_n \end{pmatrix}$$

   The right-most matrix can be simplified using the definition of Fibonacci numbers to be as desired.

(b) Prove the following: if m divides n, then f_m divides f_n, i.e., $m|n \implies f_m|f_n$.
Solution: $m|n$ iff $n = km$, so show that $f_m|f_{km}$ by induction on k. If $k = 1$, there is nothing to prove. Otherwise, $f_{(k+1)m} = f_{km+m}$. Now, using a separate inductive argument, show that for $y \geq 1$, $f_{x+y} = f_y f_{x+1} + f_{y-1} f_x$, and finish the proof.

3. Do exercise 1.6 and 1.7 in the notes.

Solution: The hint to exercise 1.6 basically solves the problem: since T_2 is connected, by adding an extra edge e we create a cycle. Trace this cycle and find an edge e' on it which is not in T_1 (if they all were in T_1, then T_1 would have a cycle!). Let this edge be e'. Clearly $T_2 \cup \{e\} - \{e'\}$ is connected (by removing e' from a cycle, we do not disconnect anything), and acyclic, and hence a spanning tree.

For exercise 1.7, let T be any MCST for a graph G. Reorder the edges of G by costs, just as in Kruskal's algorithm. For any block of edges of the same cost, put those edges which appear in T before all the other edges in that block. Now prove the following loop invariant: the set of edges S selected by the algorithm with the initial ordering as described is always a subset of T. Initially $S = \emptyset \subseteq T$. In the induction step, $S \subseteq T$, and S' is the result of adding one more edge to S. If $S' = S$ there is nothing to do, and if $S' = S \cup \{e\}$, then we need to show that $e \in T$. Suppose that it isn't. Let T' be the result of Kruskal's algorithm, which we know to be a MCST. By the exchange lemma, we know that there exists an $e' \notin T'$ such that $T \cup \{e\} - \{e'\}$ is a ST, and since T was a MCST, then $c(e') \leq c(e)$, and hence e' was considered *before* e. Since e' is not in T', it was rejected, so it must have created a cycle in S, and hence in T—contradiction. Thus $S \cup \{e\} \subseteq T$.