School of Computer Science, University of Windsor

60-254: Data Structures and Algorithms

Term: Fall 2014

Instructor: Dr. Asish Mukhopadhyay

Lab 1
Posted: 5th September, 2014
Due: On or before 14 September, 2014

Instructions:

- You are expected to finish the lab during the lab hour with help from the GAs/TAs and me. done. To help you in this, the lab will be posted sufficiently in advance so that you can think over the work to be done.
- Whether or not you finish your work during the lab hour you will have to upload your work on CLEW for record-keeping and grading before the beginning of the next lab. Create a script file as follows:

```
1. script LabName.txt
2. cat LabName.c
3. cat input.txt
4. cc labName.c
5. a.out < input.txt
6. ls -l
7. exit (DO NOT FORGET THIS STEP!!)
```

- If you are absent in a lab, you will receive 0 credit, unless there is a genuine reason, supported by appropriate documentation, in which case you will be allowed to make up the work.

Problem:

In the class we discussed an algorithm for computing the gcd (greatest common divisor) of two positive integers, m and n . We can extend this algorithm to determine integers u and v such that $um + vn = gcd(m, n)$. An algorithm for doing this is given below (you have to figure out why it works):

Step 1. Set (u', v') to (1,0).
Set (u, v) to (0,1).
Set (m', n') to (m, n).

Step 2. If (m' mod n') = 0 then go to Step 5.

Step 3. Set (temp1, temp2) to (u, v); // save (u, v)
Set (u, v) to (u', v') - (m' div n') *(u, v) ; // update (u, v)
Set (u', v') to (temp1, temp2). // update (u', v') with the saved old (u, v)

Step 4. Set r to m' mod n' ;
Set m' to n';
Set n' to r and go to Step 2.

Step 5. Output $um + vn$ and n'

Implement this algorithm in C. Make a table that outputs $um + vn$ and n' for 50 randomly generated pairs of integers (m, n) ; for each input instance, the values $um + vn$ and n' should agree. In another column, output the number times the division step is performed for each input instance (you have to set a counter in the above algorithm for this).

We argued in class that the GCD algorithm terminates in at most 2 $\lceil(\log n)\rceil + 1$ steps, where n is assumed to be the smaller of the two numbers m and n . In another column print the value 2 $\lceil(\log n)\rceil + 1$ for each input instance (m, n) , where n is assumed to be the smaller of m and n. Check that the observed number of division steps never exceed this theoretical estimate.

Comment your program carefully so that it can be read and understood. If your program is not properly commented you lose 1 point (10 points).