CSCI 1120 (Low-Level Computing), Spring 2009

Homework 4

Assigned: March 2, 2009.

Due: March 23, 2009, at 5pm.

Credit: 20 points.

1 Reading

Be sure you have read, or at least skimmed, the readings for 3/02, linked from the "Lecture topics and assignments" page[1].

2 Programming Problems

Do the following programming problems. You will end up with at least one code file per problem. Submit your program source (and any other needed files) by sending mail to `bmassing@cs.trinity.edu`, with each file as an attachment. Please use a subject line that mentions the course number and the assignment (e.g., "csci 1120 homework 4"). You can develop your programs on any system that provides the needed functionality, but I will test them on one of the department's Linux machines, so you should probably make sure they work in that environment before turning them in.

1. (10 points) Write a C program that gets a sequence of integers from the user (ending with anything that's not an integer) and then prints:

   - The smallest number entered.
   - The largest number entered.
   - The average of all the numbers entered (sum of all numbers divided by how many there are).

   Here is a sample execution of the program (user input shown in italics):

   ```
   enter some integers, anything non-numeric to end
   20
   40
   -4
   4
   ruru

   minimum = -4
   maximum = 40
   average = 15.000000
   ```

[1] `http://www.cs.trinity.edu/~bmassing/Classes/CS1120_2009spring/HTML/schedule.html`

Your program should do something reasonable if no numbers are entered (e.g., print "no numbers entered"). It also should work for any number of inputs (so you probably should *not* try to use an array to store the input).

You may (or may not) find it helpful to use constants `INT_MIN` and `INT_MAX` (the smallest and largest `int`s), defined in `limits.h`.

2. (10 points) In the early days of the Internet, a popular way of disguising text that some might find offensive was to encode it using a scheme called rot13 (short for "rotate 13"). In this scheme, all letters are rotated 13 positions (so 'a' becomes 'n', 'b' becomes 'o', 'n' becomes 'a', etc.). Spaces, digits, punctuation, etc., are not changed. (An advantage of this scheme is that if you apply it twice, you get the original text back. Think about why!)

   Write a C program that, given two command-line arguments *infile* and *outfile*, encodes the text from *infile* using rot13 and writes the result to *outfile*. It should print an appropriate error message if called with fewer than two arguments, or if either file cannot be opened.

   For example, if *infile* contains

   ```
   Now is the time for all good persons to come to the aid of their
   party.  Hello world!  1234  !@#$
   ```

   then *outfile* will contain the following

   ```
   Abj vf gur gvzr sbe nyy tbbq crefbaf gb pbzr gb gur nvq bs gurve
   cnegl.  Uryyb jbeyq!  1234  !@#$
   ```

   Hints:

   - Here is a partial function to do the required encoding.

     ```
     int encode(int input_char) {
         if (('a' <= input_char) && (input_char <= 'm')) {
             return input_char + 13;
         }
         else if (('n' <= input_char) && (input_char <= 'z')) {
             return input_char - 13;
         }
         /* add code for uppercase letters, other characters */
     }
     ```