CSCI 1320 (Principles of Algorithm Design I), Fall 2008
Homework 5

Assigned: October 30, 2008.

Due: November 6, 2008, at 5pm.

Credit: 40 points.

1 Reading

Be sure you have read chapter 6.

2 Programming Problems

Do the following programming problems. You will end up with at least one code file per problem. Every code file should begin with comments identifying you as the author and describing its purpose. It should be readable to human readers as well as to the compiler, so use consistent indentation and meaningful variable names. Feel free to cut and paste code from any of the sample programs on the course Web site.

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 1320 homework 5"). 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.

New requirement: You should also be sure your programs compile without warnings using the flags `-Wall` and `-pedantic`. You will lose points if they do not.

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
   ```

Your program should do something reasonable if no numbers are entered (e.g., print "no numbers entered").

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. (30 points) Write a C program that gets a positive odd integer `N` from the user and prints an `N` by `N` pattern of stars and spaces like the following, for `N = 11`:

```
**********************
**********  **********
********      ********
******          ******
****              ****
**                  **
****              ****
******          ******
********      ********
**********  **********
**********************
```

If the user types something other than a positive integer, the program should print an appropriate error message. If the user types an even positive integer, the program can either print an error message or use the next larger number.

You might find it useful to split your program into several functions, as a way of keeping the main program from being too complicated, and also as a way of not writing similar code over and over. Functions you might find useful in addition to the main one:

- A function that, called with a character `c` and an integer `n`, prints `c` `n` times.
- A function that, called with an integer `row`, computes how many spaces should go in row `row`.

(These are only suggestions — you may think of a way of decomposing the problem you like better.)