CSCI 1320 (Principles of Algorithm Design I), Fall 2007

Homework 4

Assigned: October 12, 2007.

Due: October 18, 2007, at 5pm.

Credit: 40 points.

1 Reading

Be sure you have read chapter 4. You may also find section 6.9 helpful.

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 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.

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. (20 points) Write a C program that defines and tests a function for converting Fahrenheit temperatures to Celsius. (So you should be able to reuse some of your code from Homework 2.) Your program should include three functions:

   - A conversion function to perform the actual conversion. This function should take one parameter, a `double` representing a Fahrenheit temperature, and return the equivalent Celsius temperature as a `double`. (So, for example, `your_function(32)` should return 0.)
   - A convert-and-print function that calls the conversion function and prints its input and output nicely — for example:

     ```
     32 degrees Fahrenheit is 0 degrees Celsius
     ```

     (Don't worry too much about printing appropriate numbers of digits after the decimal point; you can do whatever is easy, or read the man page for `printf` to find out how to get more control.)
   - A main function that calls the convert-and-print function at least four times with different inputs that you think will illustrate that the conversion is working right. (So this program is self-contained and doesn't prompt the user for anything!)

2. (20 points) Write a C program to perform a countdown using a recursive function. The program should prompt the user for two positive integers, a starting point S and a decrement D, and "count down" starting with S and decrementing by D, printing each number until a negative number is reached (which should not be printed). Output should look something like the following:

   - For S = 10 and D = 1:

     ```
     Starting point 10, decrement 1
     And away we go ....
     10
     9
     8
     7
     6
     5
     4
     3
     2
     1
     0
     Blastoff!
     ```

   - For S = 7 and D = 3:

     ```
     Starting point 7, decrement 2
     And away we go ....
     7
     5
     3
     1
     Blastoff!
     ```

   Your program should print an appropriate error message if the user enters something other than integers, or if both numbers are not positive. (Think for a minute about what would happen if D were zero!)

   You can do this problem using various kinds of loops, but to get full credit you must use a recursive function. You'll probably want one that takes two parameters, a starting point and a decrement, and doesn't return anything, but has the side effect of printing all values from the first parameter down through the last value to print.