CSCI 1320 (Principles of Programming I), Spring 2012

Homework 4

Credit: 30 points.

1 Reading

Be sure you have read chapters 5 and 6.

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 and the assignment (e.g., "csci 1320 homework 4" or "POP I hw4"). 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. (5 points) For the previous assignment you wrote a Scala program to compute U.S. income tax given taxable income. It probably involved some rather repetitive code, since the calculations for all the income tax brackets are similar (multiply by a percentage and then subtract something) but using different rates and subtraction amounts. Your mission for this problem is to reduce the amount of near-duplicate code in your program by using a function to compute the tax. This function (you choose its name) should take as parameters taxable income, the rate to multiply by, and the amount to subtract, and it should return the amount of tax. It probably makes the most sense for the function to do the whole computation, including rounding.

   To the user this program will look the same as the previous program (prompting the user for taxable income and then printing either an amount of tax or a message about using the tax table), but the code should be simpler and with less repetition.

2. (5 points) Write a Scala program that uses a recursive function to compute `b` to the `e`-th power (`b` multiplied by itself `e` times), where `b` and `e` are `Int`s and `e` is not negative. The program should prompt for `b` and `e` and print the result of computing `b` to the `e`-th power. It should print an error message if `e` is less than zero. Examples:

   - 10 to the 2nd power is 100.
   - -2 to the 3rd power is -8.
   - -1 to the 4th power is 1.
   - Any number to the 0th power is 1 (with the possible exception of 0 to the 0th power, which is probably not a sensible thing to try to compute, so you could print an error message).

3. (10 points) Write a Scala program that uses a recursive function to "count" up or down, specifying a starting value `start`, an ending value `end`, and an increment `incr`. (All three values should be `Int`s.) The function should then count starting at `start`, incrementing by `incr`, and stopping when the next value would go beyond `end`. Examples:

   - `count(1, 6, 1)` should print the values 1, 2, 3, 4, 5, 6.
   - `count(6, 1, -1)` should print the values 6, 5, 4, 3, 2, 1.
   - `count(1, 10, -1)` should print the value 1.
   - `count(4, 10, 2)` should print the values 4, 6, 8, 10.
   - `count(4, 10, 4)` should print the values 4, 8.

4. (10 points) Write a Scala program that prompts the user for lines of input, ending with a line "quit", and prints the lines in reverse order. (Yes, you can do this with a recursive function!) Sample execution (text **in boldface** is what you type; text `in typewriter font` is what the program prints):

   ```
   [bmassing@xena02]$ scala reverse.scala
   enter lines of text, quit to end
   hello world
   more text
   still more
   another line
   quit

   here are the lines you entered, in reverse order:
   another line
   still more
   more text
   hello world
   ```

   You can use the library function `readLine` to read a line of text; for example, the statement

   ```
   val s = readLine
   ```

   makes `s` a `String` that contains a line of text typed by the user.