Programming Project 9

Due: Thursday, 10/24/13 at 11:59 pm

Recursion

Recursive code is code that calls itself. This is often a tough concept and many jokes are created about this such as:

> The definition of **recursion** in the dictionary is:
>
> > see recursion

Also if you look up recursion in the index in the back of the K&R C book, one of the page numbers listed is the location of recursion in the index. Even the authors were trying to make light of a difficult subject.

One thing to recall is that recursion is based on induction. So don't forget to define your base case as well as your recursive case(s).

Often recursion is just another way to create a loop. This is often called "tail recursion".

Lab Assignment 8

For this lab, write a C program that will implement the following recursive algorithms:

- factorial
- fibonacci
- isPrime
- Greatest Common Divisor (Using Dijkstra's Algorithm)
- exponentiation

All of this code can be found on the internet. For a great page check out: http://www.arl.wustl.edu/~jst/cse/131/Notes/Recursion/recursion.html . **Of course, I don't want you just to copy the code from that page, but I want you to understand how it is working!!!**

Your program is to prompt the user for an integer value. If the user enters the value of zero, quit the program. If the user enters a value in the range from 1 to 5, perform one of the following operations based on the values given next to the name of the algorithm:

1. factorial
2. fibonacci
3. isPrime
4. Greatest Common Divisor: "GCD"
5. exponentiation

If the user gives any other value, print out a menu listing the user's options.

If the user entered a value from 1 to 5, you will then need to prompt the user for the proper number of inputs. Factorial, fibonacci and isPrime take a single integer as input. GCD and

exponentiation take two integers as input. When two input values are given, the user is to enter these values on one line of input.

Verify that the input falls into the proper range of values for the recursive algorithm. If not, print a message stating the required range for the input.

If the input values are valid, perform the algorithm and print a descriptive message showing the input and the output of the algorithm.

Recursive Definitions:

Factorial of n, written as ***n!***, is recursively defined as:

When n >= 0

- n! = 1 if n == 0
- n! = n * (n-1)! otherwise

Fibonacci of n, written as ***fib(n)***, is recursively defined as:

When n >= 0

- fib(n) = 0 if n == 0
- fib(n) = 1 if n == 1
- fib(n) = fib(n-1) + fib(n-2) otherwise

IsPrime of n, written as ***isPrime(n)***, is recursively defined with the use of a helper function as shown below. ***isPrime(n)*** itself is not recursive, just the helper function ***factorInRange()*** is recursive. Recall: an integer *n* is prime iff *n* >= 2 and *n*'s only factors are 1 and itself:

- isPrime(n) = FALSE if n < 2
- isPrime(n) = TRUE if (factorInRange (2, n) == FALSE)
- isPrime(n) = FALSE otherwise

- factorInRange (m, n) = FALSE if m >= n
- factorInRange (m, n) = TRUE if n % m == 0
- factorInRange (m, n) = factorInRange(m+1, n) otherwise when m equals 2
- factorInRange (m, n) = factorInRange(m+2, n) otherwise when m is not equal to 2

Greatest Common Demonimator of m and n, written as ***gcd(m, n)***, using Dijkstra's Algorithm, is recursively defined as:

When m > 0 and n > 0

- gcd (m, n) = m if m == n
- gcd (m, n) = gcd (m-n, n) if m > n
- gcd (m, n) = gcd (m, n-m) if m < n

Exponentiation of m and n, written as ***exp(m, n)***, has multiple recursive definitions. You must use the following recursive definition:

When n >= 0

- exp (m, n) = 1 if n == 0
- exp (m, n) = m if n = 1
- exp (m, n) = exp (m, n/2) * exp (m, n/2) if n is even **(see note below)**
- exp (m, n) = m * exp (m, n-1) if n is odd

NOTE: For the case of exp(m, n) when n is even, **don't make two recursive calls!** Make one recursive call and store the result in a local temporary variable. Then return the value in the local temporary variable multiplied by itself!

Command Line Argument: Debug Mode

Your program is to be able to take one optional command line argument, the -d flag. When this flag is given, your program is to run in "debug" mode.

When in debug mode, your program is to have each recursive function print out two messages. These messages will help show how the recursive functions work and how many recursive calls are actually made.

1. Print out a message as the first line of every recursive function that shows the name of the function and current values of all of the parameters.
2. Print out a message right before the return that shows the name of the function, the values of the parameters when this instance of the function was called and the value about to be returned. Note to print out the parameters for this call, you may need to save those values in a local temporary variable and then print out these saved values. This will be needed if the recursive function modifies the parameter(s) while executing the function.

When the flag is not given, this debugging information should not be displayed. One simple way to set up a "debugging" mode is to use a boolean variable which is set to true when debugging mode is turned on but false otherwise. Then using a simple if statement controls whether information should be output.

```
if ( debugMode == TRUE )
        printf (" Debugging Information \n");
```

Program Submission

You are to submit the programs for this lab via the Assignments Page in Blackboard.

To help the TA, name your file with your net-id and the assignment name, like:

- ptroy1LabX.c