Lab # 11

December 08, 2016

Task 1: (*Visualizing Recursion*) It's interesting to watch recursion "in action." Modify the factorial function of the following program to print its local variable and recursive call parameter. For each recursive call, display the outputs on a separate line and add a level of indentation. Do your utmost to make the outputs clear, interesting and meaningful. Your goal here is to design and implement an output format that helps a person understand recursion better.

(3 marks)

```
1  // Fig. 5.18: fig05_18.c
2  // Recursive factorial function.
3  #include <stdio.h>
4
5  unsigned long long int factorial( unsigned int number );
6
7  // function main begins program execution
8  int main( void )
9  {
10    unsigned int i; // counter
11

12    // during each iteration, calculate
13    // factorial( i ) and display result
14    for ( i = 0; i <= 21; ++i ) {
15       printf( "%u! = %llu\n", i, factorial( i ) );
16    } // end for
17 } // end main
18
19 // recursive definition of function factorial
20 unsigned long long int factorial( unsigned int number )
21 {
22    // base case
23    if ( number <= 1 ) {
24       return 1;
25    } // end if
26    else { // recursive step
27       return ( number * factorial( number - 1 ) );
28    } // end else
29 } // end function factorial
```

Task 2: Write a *recursive* function `power(base, exponent)` that when invoked returns $base^{exponent}$. For example, `power(3, 4) = 3 * 3 * 3 * 3`. Assume that `exponent` is an integer greater than or equal to 1. **(3 marks)**

Task 3: The Fibonacci series:

0, 1, 1, 2, 3, 5, 8, 13, 21, …

begins with the terms 0 and 1 and has the property that each succeeding term is the sum of the two preceding terms.

The following program implements a *recursive* function `fibonacci(n)`, which calculates the nth Fibonacci number.

```
1   /* Fig. 5.15: fig05_15.c
2      Recursive fibonacci function */
3   #include <stdio.h>
4
5   long fibonacci(unsigned int);
6
7   int main()
8   {
9      long result;
10     unsigned int  number;
11     printf( "Enter an integer: " );
12     scanf( "%u", &number );
13     result = fibonacci( number );
14     printf( "Fibonacci( %u ) = %ld\n", number, result );
15     return 0;
16  }
17
18  /* Recursive definition of function fibonacci */
19  long fibonacci( unsigned int n )
20  {
21     if ( n == 0 || n == 1 )
22        return n;
23     else
24        return fibonacci( n - 1 ) + fibonacci( n - 2 );
25  }
```

a) Write a *nonrecursive* function `fibonacci(n)` that calculates the nth Fibonacci number. Use `unsigned int` for the function's parameter and `unsigned long long int` for its return type.

b) Determine the largest Fibonacci number that can be printed on your system.

(4 marks)

Grading and LMS Submission

- Make sure that the lab engineer has graded your programs until 5 pm.
- You've uploaded the C source files in Zip format over LMS until 5:30 pm.