

CGS 3460 Computer Programming Using C, Spring 2008

Homework 4

Due Monday, March 3 2008, before 11:59:59pm

Notes

- All submissions must be done electronically via the Courseworx system linked from the website.
 - Create a separate C source file for each problem. Problem 1 must be in `p1.c` and Problem 2 in `p2.c`.
 - Use `tar` to combine all the files into the file `H4.tar`. To do this, type `tar -cvf H4.tar p1.c p2.c` on the command line. You must only upload `H4.tar` to Courseworx.
 - Once you've uploaded the file, download the file and untar it using the command `tar -xvf <filename>`. Display each extracted file to verify your programs are intact.
 - Submit C source files *only* (files with `.c` extension). We will compile and run them.
 - The first three lines of each C source file must contain your Full Name, UFID and Gatorlink ID as comments.
 - Make sure that your code compiles and runs correctly on one of the following CISE machines: `sand.cise.ufl.edu`, `rain.cise.ufl.edu`, `shine.cise.ufl.edu`, `bay.cise.ufl.edu`.
 - When obtaining input, be sure to prompt the user appropriately.
1. You wish to write a bunch of functions to help with String handling. Here is a description of the functions:
 - `fgets_generalized` which takes in same arguments as `fgets` except for another argument (which is the last argument), a single char variable (call it `term`), which will be the termination character. That is, it will read in input, until `term` is encountered. This could be over multiple lines, unlike `fgets` which reads in only a single line. The termination character must also be present as the last character of the string.
 - `word_count` which given a String (terminated with a `'\0'`), reports the number of words in the string. A word is defined as a sequence of characters that does not contain a blank, tab or newline.
 - `unblank` which takes in a String (terminated with a `'\0'`), replacing each sequence of one or more blanks by a single blank.
 - `to_lower` which takes in a String (terminated with a `'\0'`), and replaces all uppercase characters with their lower case equivalents.

In the `main()` function, obtain the termination character from the user and then use `fgets_generalized` to get the string from the user. Then demonstrate the result obtained by executing each of the functions you defined (as above).

You may use any function in `string.h` and `stdio.h` in your program.

2. Write a *recursive* function which takes in one `int` argument `i` and generates the i^{th} Fibonacci number. Recall (from the exam) that Fibonacci numbers are a sequence of numbers, for which the n^{th} number is given by

$$F(n) := \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ F(n-1) + F(n-2) & \text{if } n > 1. \end{cases}$$

For example, the first ten Fibonacci numbers are 0, 1, 1, 2, 3, 5, 8, 13, 21 and 34.

Use this function in a C program which allows the user to enter a number i and prints the i^{th} Fibonacci number.