

CS 385  
Homework Assignment 0  
Due by Tue, Sep 2, 2008.

Prof. Dietrich

You are expected to learn enough of the C programming language to implement the algorithms covered in class. There is no way to learn a programming language other than by *writing programs*. C is a language notable for its very efficient compiler, so that experts prefer to use it when speed is paramount. However, it is possible to write very concise programs whose purpose is quite obscure, even to experts (aka the Obfuscated C contest). So aim for clarity.

A very good book on C is: *The C programming language*, by Brian Kernighan and Dennis Ritchie, the inventors of C.

You are encouraged to read it. It's in paperback, and is on the recommended book list.

The process of writing a program in C is the following.

- Using a text editor, you write the program, say `foo.c`. (Use the extension “.c” for C programs. Use a separate directory for each program, e.g. `homework1`. The extension “.cc” is used for C++ programs.)
- Compile the program, by running a C compiler. The compiler translates the C program into a machine executable program.
- If you get compiler-time errors, repeat the process.
- If you get no errors, run the program on several inputs, to verify it behaves in the expected way. If the program produces run-time errors, go back and check the program.

A sample program.

```
#include <stdio.h> /* this is a library file containing
                    functions for input/output */

int main() /* every program needs to have a function
            called 'main'.
{
    printf("Hello, world!\n"); /*printf is a library function*/
                               /* \n is the newline character */
```

```
return 0: /* main returns an integer */
}
```

Note that comments are contained in “/\* ... \*/”.

Now, save this file, as “first.c”, say. To compile, execute

```
% gcc first.c -o first
```

Here, % is the command prompt. It may vary in your shell. gcc is the C compiler on our Unix machines. This command says, compile the program named first.c and call the resulting executable file “first”. As an alternative, you may use cc, which should be equivalent.

If you haven’t made an error, you may now run the program by executing

```
% ./first
```

The expression “./” denotes the current directory. Now repeat the process, and execute:

```
% script hw1
% cat first.c
% cc first.c -o first
% ./first
% exit
```

In summary, this creates a typescript (look up “man script” on Unix) of a terminal session. You display the program with cat, then compile it, execute it, and then close the typescript via exit. You should remain at the shell prompt. There will be the current time and date displayed both at the beginning and the end of the typescript.

## The assignment

Learn enough about the syntax of C to write a program, which must be called pow.c, which does the following.

It takes from the user two positive integers,  $n, p$ , and outputs the integer  $n^p$ , the  $p$ -power of  $n$ . You may recall that  $n^p = n * \dots * n$ ,  $p$  times. Or, inductively,

$$\begin{aligned} n^0 &= 1 \\ n^{p+1} &= n^p * n, \end{aligned}$$

where  $x * y$  is the product of  $x$  and  $y$ . Here is a typical run of the program.

```
% pow
Enter two (small) postive integers:
12 3
12 to the power 3 is: 1728
```

The program prompts the user for two integers. In this case, the user enters 12 and 3. The program then responds with the result of computing  $12^3$ .

For beginners, this is NOT a trivial task. Aside from the arithmetical part, you must learn how to obtain the integers from the user, and how to output the result. At this time, don't worry about "large integers", say those requiring more than 8 digits.

Good luck. Bring a printout of your typescript to recitation on Sep 2.

## Extra credit

Enhance your program to compute the function

`powmod(n,p,m)` which returns  $n^p \pmod{m}$ .

[In C, the percent sign is used for mod, e.g., `x % m` is  $x \pmod{m}$ .]

Recall, for positive integers  $x, m$ ,  $x \pmod{m}$  is the remainder of  $x$  when divided by  $m$ ; for example,

$$\begin{aligned} 12 \pmod{3} &= 0 \\ 12 \pmod{5} &= 2 \\ 12 \pmod{15} &= 12. \end{aligned}$$

You should use the facts that

$$\begin{aligned} (x * y) \pmod{m} &= ((x \pmod{m}) * (y \pmod{m})) \pmod{m} \\ (x + y) \pmod{m} &= ((x \pmod{m}) + (y \pmod{m})) \pmod{m}. \end{aligned}$$

Compute  $2^{12345} \pmod{1775}$ . Note that your `pow` program will get the wrong answer if you ask it to compute  $2^{12345}$ .