CSCI 247 Computer Systems I
Fall 2016
Laboratory Exercise 1

Objectives

1. Gain some familiarity with the C programming language.
2. Practice compiling and running C programs.
3. Gain some experience with characters integers and strings in C.

Submitting Your Work

Submit your C program files as via the **Lab Exercise 1 Submission** item on the course web site. You must submit your program by 5:00pm on the Friday following your scheduled lab session.

Introduction

Your task for this lab is to write two C programs: one which can work as a simple calculator for decimal numbers, and another which will work for hexadecimal numbers.

Decimal Number Calculator

For this program you start with an existing, but incomplete program `calc.c`, which is available in the Lab1 folder on the course web site. The existing program uses two functions `numval()` and `docalc()` which are declared, but not defined in `calc.c`. Your task is to complete the definitions of those functions. Here are the declarations of the functions:

```
int numval(char *word)
```

This function returns the number represented by the string parameter `word`. If the string `word` does not correspond to a valid integer number, `numval()` must return a negative integer.

Note: There are functions in the C library, `sscanf()` and `atoi()`, among others, which can do this task, however, you are **NOT** permitted to use them. You may only use the library function `strlen()`, which will tell you the number of characters in a string. In order to use that function you should include the header file `string.h`.

```
int docalc(int first,
           int second,
           char operator)
```

This function returns the result of applying the `operator` to the integers `first` and `second`. The `operator` may be +, -, * or /, representing addition, subtraction, multiplication and integer division, respectively. If an invalid operator is received, the function must abort the program by calling the function `outahere()`.

After modifying `calc.c`, compile it to produce the executable file `calc`:

```
gcc -Wall -o calc calc.c
```

Then you can run the program, as shown in the examples below:

```
./calc 23 + 78
23 + 78 = 101
./calc 23 + cat
Usage: ./calc number op number, where op is +, -, * or /
./calc 23 % 78
Usage: ./calc number op number, where op is +, -, * or /
```

Hexadecimal Calculator

Copy your program `calc.c` to make a new file `hexcalc.c`

```
cp calc.c hexcalc.c
```

Now modify the `numval()` function in `hexcalc.c` to convert a string representing a hexadecimal number to an integer. The `numval()` function must accept digits '0' through '9', lower-case letters 'a' through 'f', and upper-case letters 'A' through 'F'.

You may display the result of the calculation in hexadecimal by using the `%x` or `%X` format in the format string for `printf()`.

Compile and run the `hexcalc` program:

```
gcc -Wall -o hexcalc hexcalc.c

./hexcalc 3E + ff
3E + ff = 13d
```

Submit your files via the course web site

Submit both `calc.c` and `hexcalc.c`.