CSE 320 **Fall 2018**

Computer Project #5

Assignment Overview

This assignment develops familiarity with the C programming language, the "gcc" compiler, number systems, and twos complement representation.

It is worth 40 points (4% of course grade) and must be completed no later than 11:59 PM on Thursday, 10/4.

Assignment Deliverables

The deliverables for this assignment are the following files:

`proj05.makefile` – the makefile which produces "proj05"
`proj05.support.c` – the source code for your conversion module
`proj05.driver.c` – the source code for your driver module

Be sure to use the specified file names and to submit them for grading via the CSE handin system before the project deadline.

Assignment Specifications

An input/output library in a programming environment will include a method for converting from external representation (character strings) to internal representation (bit patterns). For example, function "scanf" in the C standard library allows the user to enter a character string representing a value, converts it into internal representation and stores it in a variable.

Consider the following example:

```
int A;
scanf( "%d", &A );    // Assume the user enters 25 at the keyboard
```

Function "scanf" accepts the character string "25" from the keyboard, converts it into the two's complement representation of the value twenty five and stores those bits (`00000000000000000000000000011001`) in memory as the contents of variable "A".

1. You will develop a C module which supports the conversion of integer values between external representation and internal representation. The module will consist of function "convert" and any other supporting functions you choose to develop. The interface to the module is:

```
int convert( const char[], int, int* );
```

The first argument is the address of an array that contains the external representation of the value which is to be converted.

The second argument is the value of the base in which the first argument is represented. Valid bases range from 2 to 36 (inclusive).

The third argument is the address of a scalar where the function will store the equivalent internal representation of the first argument.

Function "convert" will return the integer value 1 if the conversion is successful, and the integer value 0 otherwise.

The external representation of the value to be converted (the first argument to "convert") is a null-terminated sequence of ASCII characters. It is composed of three disjoint substrings: a substring of zero or more white space characters (blanks, tabs and newlines), a substring which represents the specified value (called the subject substring), and a substring of one or more unprocessed characters (including the terminating null).

The subject substring is defined as the longest possible substring which is of the expected form. It begins immediately after any leading white space, and terminates with the first invalid character. The subject substring may not be empty.

The subject substring represents a signed value. A negative value is indicated by the character '-'; a positive value is indicated by the character '+' or no sign character.

In the subject substring, alphabetic characters are used to represent digits for bases greater than 10. Both upper and lower case letters are valid.

2. You will develop a driver module to test your implementation of the conversion module. The driver module will consist of function "main" and any additional helper functions which you choose to implement. All output will be appropriately labeled.

Assignment Notes

1. Your conversion module and your driver module must be in separate source code files.

2. Your source code must be translated by "gcc", which is a C compiler and accepts C source statements.

3. You must supply a "makefile" (named "proj05.makefile"), and that makefile must produce an executable program named "proj05".

4. Your driver module may not be written as an interactive program, where the user supplies input in response to prompts. Instead, your test cases will be included in the source code as literal constants.

5. You may not call any C library functions from your conversion module. That is, you may not call functions such as "printf", "scanf", "atoi", or "isspace" from "convert" or any other function in your conversion module.

6. Your conversion module must convert between integers and characters without using any C library functions. The equivalent integer value of a character from the set {'0'..'9'} representing a decimal digit may be obtained by:

```
char ch;
int value = ch - '0';
```

The equivalent integer value of a character from the set {'A'..'F'} representing a hexadecimal digit may be obtained by:

```
char ch;
int value = (ch - 'A') + 10;
```

A similar strategy can be used for characters from the set {'a'..'f'} representing a hexadecimal digit.