Assignment Overview

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

It is worth 30 points (3% of course grade) and must be completed no later than 11:59 PM on Thursday, 10/5.

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 library 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 internal representation (bit patterns) to external representation (character strings). For example, function "printf" in the C standard library allows the user to display a character string representing the contents of a variable.

Consider the following example:

```c
int A = 0x5d;       // A: 00000000000000000000000001011101

printf( "%d", A );
```

The declaration places the twos complement representation of the value 5d base 16 into variable "A" (00000000000000000000000001011101). The call to "printf" places the characters "93" into the output buffer.

1. You will develop a library module which supports the conversion of signed integer values between twos complement internal representation and external representation. The library module will consist of function "decode" and any additional helper functions which you choose to implement. The interface to the module is:

```c
int decode( int, const char[], char[] );
```

The first argument is the value which is to be converted into external representation.

The second argument specifies the base into which the first argument is to be converted. Valid specifiers are "bin" (binary), "oct" (octal), "dec" (decimal), and "hex" (hexadecimal).

The third argument is the address of an array where function "decode" will store the external representation of the value as a null-terminated sequence of ASCII characters. If the conversion is successful, that representation will be in the form "+N" or "-N", where "N" is the sequence of digits representing the value. Otherwise, it will be the empty string.

Function "decode" will return the integer value 1 if the conversion is successful, and the integer value 0 otherwise.
2. You will develop a driver module to test your implementation of the library 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 library 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 library module. That is, you may not call functions such as "printf", "scanf", "atoi", or "isspace" from "decode" or any other function in your library module.

6. Your library 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:

   ```c
   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:

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

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

7. Consider the following, which shows the character strings produced by a series of successful conversions.

   ```c
   int A = 127;
   int B = -6;
   char C[80];
   
   decode( A, "dec", &C[0] );  ==>  C contains "+127"
   decode( A, "hex", &C[0] );  ==>  C contains "+7F"
   decode( A, "bin", &C[0] );  ==>  C contains "+1111111"
   decode( B, "dec", &C[0] );  ==>  C contains "-6"
   decode( B, "oct", &C[0] );  ==>  C contains "-6"
   decode( B, "bin", &C[0] );  ==>  C contains "-110"
   ```

   For unsuccessful conversions, the character string should be empty.