# Computer Science 304
# Computer Organization
# Spring 2025
# Assignment 6 – Base Calculator

**Due: Friday, 3/28/2025, 11:59 p.m.**

For this project, write a program in C that extends the program you wrote for Assignment 3. Once again, the program will display values in various base representations, including binary, based on the current value. It will also perform various bit-wise and arithmetic operations involving the value. The program's execution is driven by the following menu:

```
**************************************
* Base Values:         Input Mode: Dec *
*   Binary  :  0000 0000 0000 0000     *
*   Hex     :  0000                    *
*   Octal   :  000000                  *
*   Decimal :  0                       *
**************************************

Please select one of the following options:

B  Binary Mode            &  AND with Value         +  Add to Value
O  Octal Mode             |  OR  with Value         -  Subtract from Value
H  Hexidecimal Mode       ^  XOR with Value         N  Negate Value
D  Decimal Mode           ~  Complement Value

C  Clear Value            <  Shift Value Left
S  Set Value              >  Shift Value Right

Q  Quit

Option:
```

Note that the value in various base representations and the menu are displayed as in the previous project. Options for the menu are also entered/accepted as before.

Since C does not have a format for reading or printing values in binary format, the program must convert the short value to or from a character string containing the binary digits.

For input, the user may enter a value in binary (with no spaces) of any length up to 16 bits. In the program, this input should occur along with other base types in **`get_operand()`**. Note that **`get_binary_op()`** will need to be called from this function so that the binary string can be converted to a short value and returned from the function.

For output, the **`val`** must be converted to a binary string using **`convert_to_binary()`**. Spaces must be inserted after every four binary digits for readability.

For consistency, declare all binary strings as user-defined type **`bin_str`**, which is a character string of size 20 (to hold the 16 bits, the string terminator character, and spaces, if necessary).

For the new menu entries, **&**, **|**, **^**, **+**, and **–**, the program will request an operand (in the current mode) from the user to perform the corresponding operation.

For the **+** and **–** operators, a message should be printed if positive overflow or negative overflow is detected (though the operation will still be performed).

When either shift operation (**<** or **>**) is selected, the user will be asked to enter the number of bits to shift (in decimal). Detection of overflow due to shifting is not required.

Note that no additional operand is needed for the **~** and **N** options.

All operation results should be reflected in the current value.

Your code must be modular, written with appropriate formatting and structure, contain comments (including a header), and use functions, as before. Include functions from the previous assignment, plus the following (but no others):

```
unsigned short get_binary_op (bin_str bin)      // convert bin str to short; return value
void convert_to_binary (short val, bin_str bin) // convert val to binary str for output
void add (short *val, char mode)                // call get_operand to get value in mode
                                                //  to add to val; detect pos/neg overflow
void subtract (short *val, char mode)           // similar to add, but subtract
```

All code should be submitted in a source file named **calc.c**, along with an accompanying **calc.h** file to hold the **bin_str** type. For the **calc.h** file, use header guards to prevent the file from being included more than once.

Your program can be compiled with the command:

```
gcc calc.c –o calc
```

..and run with the command:

```
./calc < calc_in.txt
```

Don't forget to test your program on the CS department computers.