Description: This project consisted of 2 programs involved with converting between different number formats. Calc, the first one, consisted of simple mathematical arithmetic using decimal, octal, binary, and hex. Format, the second one, consisted of converting a given binary string input into either a float of a decimal.

Calc algorithm: For calc, there are 4 arguments that are presented to the program. The operator, number 1, number 2, and the desired base. My programs takes and stores the information into variables, checking for errors along the way to make sure there aren't any discrepancies. It converts both numbers into integers types using the functions in the header files so that the arithmetic can be done. There are three arithmetic symbols that are supported: addition, subtraction, and multiplication. After it does all the arithmetic, it then converts it back into an ascii string of the desired base also using the functions in the header function. For negative numbers, I implemented it as having a -1 times whatever number it was given. Also when printing out the negative number, instead of going through the trouble of adding a negative number back into the string I just added it before the printed variable.

Format algorithm: For format, there are two arguments that are presented to the program. A 32 bit binary string and the desired base. Checking for any discrepancies in the arguments, it goes on converting to the desired base. For int, using the a function in the binary header file, it becomes quick and painless. For float, a union was needed to be able to copy the binary bits from the ASCII to a integer and finally into the float. If '01111111100000000000000000000000' or '11111111100000000000000000000000' was inputted, the program defaulted to infinity since no calculation had to be done there. Any other input, using floatToASCII, the float number is converted to ASCII. One trouble that I have come across was when using the floatToASCII function, weird characters were showing up in the final resulting string. Adding a null byte at the end of the string (compensating for the '-' symbol) solved this issue. The resulting string is printed out.