For my calc program I transformed both strings into a struct which held the base 10 equivalent of the input, whether the number was negative or not. Validation of the string inputs was pretty much like my tokenizer program, comparing each character in a string to make sure it was a valid character for that indicated input(B,O,X,D). The strings after validation were converted to base 10 equivalent and stored inside the struct. All arithmetic was performed in base 10 int which allowed to just call the addition, multiplication and subtraction operators. The answer stored in an int was then converted into a string based on what the user wants outputted (B,O,X,D).

The biggest difficulties was detecting if an input or answer was longer than 32 bits. My first method for string conversion was to iterate through the string and add onto an int result based on the type of input and what was the current char. I.E. result = result *2 + (arr[i] – '0') (the 2 is for binary, I would use 8 for octal, 16 for hexadecimal and 10 for regular decimal). If at any point my result became smaller than my previous result, I knew an overflow had occurred. I would tack on a negative after I found my result (my struct held a negative indicator). However my only problem was -2147483648 which is the smallest 32 bit integer, working in positive, I can only goto 2147483647. To handle this kept my result negative if the number was indicated to be negative. For my answer, I checked certain conditions where my answer should be greater or less than one of the input numbers.

I used two structs (for each input number), one int to hold the answer and a char* that held the answer. I used char* to point to one string as I edited allowing for less space problems. My space analysis would be mostly on the string arrays that held each input and the answer. For my Big-O time analysis, O = n, where n is the number of characters entered. I iterate through the two stings many times, my limit would be when n chars gives an int greater than 32 bits. The longest string would be a 33 binary bit string.