Adit Patel

Programming Assignment 2: Data Representation and Computer Arithmetic

Calc

My program adds, subtracts, and multiplies two numbers with different or same bases. These bases can be decimal, binary, octal, or hexadecimal. My thinking for the program was essentailly turning everything into a decimal, doing the operations on them, and then converting everything back to the output base. I had one function that turned the inputs num1 or num2 into an integer representation. The reason I did this is because we did not have to worry about arbitrary size, since the professor said that. This being said, everything would fit into 34 bits (because of the '-' and the output base). In my function to turn everything to int, I had a whole bunch of if/elses for each base. I first converted all the actual numbers (everything except the '-' or base) into a string. From there, I converted that string to an int. To get the characters into a number representation, I used - '0'. After I got the int, I checked if there was a '-' in the input. If there was, I multiplied the output by -1, and returned that.

In my main method, I checked the operations and then did the computation. I did this by converting both inputs into a decimal, then either adding, subtracting, or **multiplying**. After I got the result from that, I had an another if/else block to convert it back to the output base. The functions I used in this block were toBinary, toOctal, toHex, and toDecimal. In each of these, I did a + '0' to convert a number into a char. In toBinary, toDecimal, and toOctal, I went from right to left, and kept on % (base) then adding it to the output. However in toHex, i took a similar approach but had to account for different things. I did %16 to get the digits, but if it was less than 10, I had to add 48, but if it was greater than 10, I had to add 55 to get the letter representation. I then added this backwards to an character array, did a strcpy to make it a char *, and then outputted that char *.

Because in toBinary, toDecimal, and toOcatal, I went from right to left digit by digit, I had to make a helper function called reverse to reverse this so it comes out as left to right. I also had to make a helper function for pow, since we couldn't include the math.h.

Big O:

Converting everything to decimal takes O(n) times to finish because it looks each character in the input and converts it to the int. My toHex, toBinary, and toOctal all take 0(n) times too because it goes character by character as well. My toDecimal takes O(log n) because the decimal is being divided by the output base (10) until it reaches 0. My reverse takes O(n) times because it goes through the string.

Space Analysis:

I malloced the exact amount of space needed for each arguement plus - and base. Worst case was 32 bit so i malloced 32 sometimes 34 bits.