Michael Wexler

Calculator Readme

Essentially, the overall design/implementation of the project is as follows:

Common.c contains all functions for converting bin, oct, or hex strings to decimal form. It also contains functions for converting decimal numbers to bin, oct, or hex strings. My add.c, sub.c, and mult.c each contain a main method, which processes the input and calls respective conversion functions, and then displays output. All input numbers are first converted into decimals. The respective operation is performed. Then the output is converted from this decimal to the output base that is wanted. Once again, all of these conversions use the functions from common.c. Common.h contains all global constants and function headers.

Algorithms for conversion:

To convert from either oct or hex to decimal, the algorithm was:

Loop through string. For each digit encountered, raise 8 or 16 to the power corresponding to that place, and add that to a running total.

To convert bin to dec:

Loop through string. If you encounter a 1, add 2 to that respective power to the running total.

To convert dec to any of the other bases:

Each time, while the number is greater than 0, divide (integer division) by the respective base, and take the remainder and output it to the beginning of an accumulating string.

Space and Time Analysis

We will analyze only the conversion methods, since the other input/output operations happen in O(1) time (and the actual add/mult/sub operation also occurs in O(1) time).

Let us first analyze the conversion of a non-decimal base to a decimal base:
Loop through array (n digits), and each time add a power of the base to a running total. This is O(n * c) = O(cn) = O(n). Thus, this operation is linear (assuming raising to a power is a constant operation). Space-wise, the calculation could involve a decimal up to roughly 5 x 10^9, which is stored internally as a 32-bit number, which takes 4 bytes.

Let's analyze now the conversion of a decimal base to a non-decimal base:

Each time, divide the number by the base. Do this until the number is equal to 0. Note that the number of divisions by a base b of a number x to reach 0 should be n = log b x. Each division will also involve a constant time calculation of remainder, and output of remainder to string. Therefore, overall this conversion is O(log n), where n is decimal inputed. If this is wanted in terms of the number of digits d in the decimal, then the runtime would be O(log n) = O(log 10d) = O(d log 10) = O(d). So the runtime is linear in terms of the number of digits. Space-wise, the storage of a non-decimal base as a string could take up to 9, 12, or 33 bits, depending on the base.

Challenges encountered

I did not encounter any challenges during the course of this assignment.