calc

calc is a program designed to do basic arithmetic on ARBITRARILY LARGE integers. calc accepts numbers of base 2, 8, 10, and 16 and performs addition, subtraction AND both multiplication and division. calc accepts arguments of the form

```
./calc <op> -?(b|d|o|h)<operand1>  -?(b|d|o|h)<operand2> <base>
```

calc prints the result of the operation on the next line in whatever base the user has specified.

Data Structures

In order to store arbitrarily large integers I used a struct BigInt which is comprised of

```
struct BigInt{
int k;
int *array;
}
```

I converted the string representation of a number into an integer array of digits, representing each digit of the decimal form of the number. This approach allows me to store any integer with theoretically a max of 2^32 decimal digits on a 32 bit computer. More if on a 64 or 128 bit computer and so on.

After each operand has been stored in a BigInt structure I can then perform arithmetic.

Algorithms/Big-O Analysis

Addition, subtraction, multiplication, and division were modeled by algorithms for hand done arithmetic you learn in grade school.

For addition, if n is the amount of digits of the largest operand, and one action is adding two digits together, then it takes n actions to complete an addition of two BigInts. So O(n).

For subtraction, if n is the amount of digits of the largest operand and one action is subtracting one digits from another, then it takes n actions to complete a subtraction of one BigInt from another. So O(n).

For multiplication, if n the amount of digits in one operand, and m is the amount of digits in another operand, and one action is multiplying a digit to a digit, then it takes m * n actions to multiply two BigInts together. So O(mn).

For multiplication, the algorithm is based on long division by hand. It also has a O(mn) run time for the same reasons as multiplication.