CPL Assignment2

Implementing Big Int

I wrote a separate function for addition, subtraction and multiplication of 2 big ints.

To store the big int, I stored each digit in a string. I used a string instead of an array because I was planning on storing each digit in each element of the array and if I used integer arrays then I would be wasting quite a lot of memory which would be located for a normal integer. Since characters take up lesser number of bytes than an integer, I decided to store each digit as a character.

The big-endian method, which is the most significant digit is stored at the smallest index of the array, which in this case is an array of characters, string.

With each of the functions I first converted each character to an integer and then used the integer version of them to do the calculations. To convert them, I just subtracted the ASCII value of the character in the number given with the ASCII value of 0.

I used the logic that we use on paper to implement all of these functions.

With the addition function I first added the digits and the carry which is initialized to 0, which was stored as an integer. If this value was greater than 10, then the digit to be stored would be the reminder of the value divided by 10, and the carry would be the value divided by 10. I keep adding the carry with the next significant digit.

With the subtraction function I used another function to tell me which of the numbers are bigger. If the first number entered has more digits than the second, I returned 1, and 2 if the second number had more digits. If both of them were equal in size, then I started to check which of the digits were larger starting from the most significant digits.

Once I found out which one was bigger, I wrote 2 functions. One function was to always output the positive value of the subtraction between 2 numbers, no matter how they were entered. The second function can return a negative or a positive number based on the order of which the numbers are entered.

The logic for both these functions are the same. If the digit of the bigger number is smaller than the corresponding digit of the smaller number, then increment the bigger numbers digit by 10, and decrement the next significant digit of the bigger number, or else just subtract the smaller numbers digit from the bigger numbers digit.

With the multiplication function I also used the same method that we do on paper. I multiplied each digit with each of the digit for the other number entered and if a carry is generated then it was added to the next significant digit very similar to addition. But, instead of adding all the digits at the end like how we do it on paper, I added the numbers to their respective indexes while multiplying the digits. I determined the indexes to be added by considering one’s place as 0, 10’s place as 1 and so on, so if we multiply one’s digit with the other numbers ones digit, the resultant should be stored in the ones place of the resultant string, which can be taken as 0 + 0. The same way if we multiply a 10’s digit with a 1’s digit, then the resultant digit will be in the 10’s place which can be taken as 0+1. Using this logic I implemented the multiplication function.