Istvan von Fedak
Lab3
ECE218
10/9/2016

Class Design:

My class is composed of the following seven fundamental parameters: a short pointer (*data) that points to an array of shorts, a length integer that keeps the track of the length of the populated array, a space integer that keeps track of the amount of space in the unpopulated array (total array size/capacity), an expandSize integer that is used to determine how much extra space will be added if the expand() function is called, a multiplier integer that multiplies the expandSize integer for faster incrementation of space, a timesMultiplied integer that keeps track of how much space is in the array and a compressed boolean that indicates if the space of the array is equal to its length (handy when wanting to conserve memory space).

Furthermore other private functions are used to assist addition, subtraction and multiplication. Compress as its name implies compresses the BigInteger making its space and length of equal size. Expand does the opposite of compress and it adds an expandSize*multiplier amount of space to the array. addToIndex adds a short with a given index to the BigInteger data. Borrow aids subtraction and its used to “borrow” 1 short from a given index. compare is a function mostly used for subtraction even though I tried to unsuccessfully implement it in division.

Algorithms used for increment, addition, multiplication and subtraction

increment:
- This algorithm uses recursion to traverse the entire array
- It has a parameter index that increments with each recursive call allowing the algorithm to traverse the entire array if needed
- special cases:
  - if the bigInteger is uninitialized then the functions sets it equal to one
  - if the bigInteger is already initialized and the incrementation of the current number in the given index of the array is less than or equal to nine, then the number in the given index is equal to itself plus one.
  - if the bigInteger is already initialized and the incrementation of the current number in the given index of the array is greater than nine, then this means that the integer is equal to 10.
    - Because the integer is equal to 10 we need to set the integer in the current index equal to 0 and increment the array one index higher, recursively calling the function.
    - Furthermore, if the index where the number being incremented is greater than the space in the BigInteger, the amount of space is expanded.

Addition:
- This algorithm uses recursion to traverse the entire array if needed
- We assume that the given integer given to add is already a populated integer
- If the data in the integer you're adding to is NULL, then the integer becomes equal to the integer being added
- Since this is a recursive function, the conditional statement allows the recursion to continue as long as the index is less than the length of the integer being added.

    - The reason behind this conditional statement is to allow the algorithm to traverse the entire length of the integer being added.
- If the length of the integer being added is greater than the base integer, then the length of the base integer is incremented.
    - The base integer's length is incremented to the length of the integer being added plus one incase the extra space is needed to be populated in future calculations.
- If the length of the integer being added is less than or equal to the base integer, then the length of the base integer is incremented to its length plus one.
    - In case the extra space is needed to be populated in future calculations.
- If at the given index of the array both numbers summated are less than or equal to nine, then the number is equal to to the sum of both numbers
- If at the given index of the array both numbers summated are greater than nine, then the current index of the base integer is set to zero and the number in the following index of the array is incremented.
    - This is the same as adding 10 to the array in the given index
- After all of that is done, the function is calls itself again incrementing the index.

Multiplication:
- The multiplication algorithm that I used uses recursion and an integer array that keeps track of the original value of the base integer
- Furthermore, I assumed that the length of the integers being multiplied was not going to be greater than both of them combined.
- Since this is a recursive algorithm, the entire length of the array being multiplied needs to be traversed.
- For this reason a boolean statements needs to check if the length of the base number is greater than its index.
    - If it isn't then the function will not call itself again breaking the recursion
- The approach I took to multiply the multiplicand, was to multiply it by all of the numbers in the multiplier(I hope that makes sense I do repeat "multi" a lot). I achieved this by using a for loop that traverses the entirety of the multiplier.
    - If the product is less than or equal to 9 then the index of the multiplicand is equal to the product
    - If not, then the remainder of the product decided by ten is added to the current index plus the index of the for loop.
        - This is because the index is used as a "weight" on the number(10*index).
    - Furthermore the remainder-less product divided by 10 is added to the current index plus the index of the for loop plus one.
        - The reason why I added a plus one to the sum of index' is because the product is one level higher in the exponent since it was greater than 9.
    - after that is done, the function is called recursively and the index of the multiplicand is increased by one.

Subtraction:
- This algorithm once again uses recursion.
- If first checks for three things.
    - If the integer array being is larger than the base integer array
        - in which case it is negative and returns -1 and an error message.
    - If the integer array is equal in size as the base integer array
        - in this case it compares the two arrays

    - If they are both equal to each other, then the base array is zero
    - If not then it treats it the same way as if the base's array length was greater than the integer array being subtracted
- If the integer array is less than the base array:
    - then it compares the result from subtracting the integer from the base integer (in the current array index)
        - If the result is greater than zero, then the base array's integer at that given index is set equal to the result
        - if it is equal to zero it checks to see of the index plus one is still less than the base array's length
            - if it is then it sets the integer in the current index of the base array equal to zero and reduces the length of the array by one.
            - If not, then he base array's integer at that given index is set equal to the result.
        - If the result is negative, then a value of ten is borrowed (one is subtracted from the number with the index plus one) and added to the result.
            - If the result is still negative that means that the base integer array is smaller than the subtractor array making it negative
                - in this case the base arrays value is set to negative one with a length of one and an error message is displayed to inform the user the arrays are negative.
            - If the result becomes positive, then the result is added to the base array.
        - Then the function is called again recursively and increments the index by one

Division:
- I tried implementing long division.
- My approach was to be able to subtract the devisor from the base number and add a certain amount of weight with the numbers position in the temporary array.
- If you have any hints on how to do it I'm open to suggestions.

Compiling instructions.
- Include both the BigInteger.h and BigInteger.cpp in the folder you're going to compile your main with. Initialize the numbers using a constructor that takes a string as a parameter (ex: BigInteger b1("1234");). You can use either the print function or cout to print the BigInteger. Other than that its like a normal library. You can always email me asking me for more in-depth compiling instructions if necessary.