DAA Assignment – 1
01FB15ECS066

The following is my approach towards creating the C library for integers of arbitrary length.

The numbers are stored in the form of a character array, preceded by the sign of the number. This array, along with the length of the number, is stored in the form of a structure “intal”.

The following functions are implemented on the integers of arbitrary length“intal”:
(a) Add two integers of arbitrary length.
(b) Subtract two integers of arbitrary length.
(c) Multiply two integers of arbitrary length.
(d) Divide two integers of arbitrary length.
(e) Exponentiation limited to positive power.

WORKING OF THE FUNCTIONS:

(a) ADDITION
and
(b) SUBTRACTION

Done using “add_intal” and “subtract_intal” respectively. They are caller functions which choose the necessary function to call (“add” or “subtract”) based on the signed inputs. The called functions “add” and “subtract” do addition and subtraction respectively without considering the signs of the inputs.
Addition in “add” is done from right to left of the strings, using the carry bit. The length of the new string is set as 1 more than the length of the longer string. The sum is given as (char from 1st str + char from 2nd str + carry). If the sum of two single digits is greater than 9, carry is set to 1 while the least significant digit is placed into the new string. If the new string at the end has a preceding zero to the answer, it is packed so as to not include the 0.
Subtraction in “subtract” is done from right to left of the strings. The length of the new string is assumed to be equal to the larger between the two given strings. If the difference between two characters goes below 0, the preceding character of the first string is decremented and 10 is added to the difference we got, thus mimicking borrow. The new string, most probably will have preceding zeros, thus the string is packed up to take care of the preceding zeros.
Both “add” and “subtract” do not consider the signs of inputs.

(c) MULTIPLICATION

Done using the function “multiply_intal”, which is a calling function that takes care of the sign of the output and also the way input is to be given to the called function “karatsuba”, where the first parameter must be greater than the second parameter. This calling function also takes care of the leading zeros to the answer.
Multiplication in “karatsuba” is done by implementing the Karatsuba Algorithm, which is included in our course and thus, details are not given here.

(d) DIVISION

Done using the function “divide_intal”, which is the calling function that takes care of the sign of the output and the leading zeros in the output. This function calls the function “divide”.
Division in “divide” is implemented in the form of recursive subtraction. Division is done without considering the sign of the input, which is taken care by the calling function. It performs integer division, giving the floor value. If the second parameter is zero, it returns an intal with NULL.

(e) EXPONENTIATION

Exponentiation is done by calling the function “pow_intal”. It does exponentiation by repeated multiplication. If the second parameter is -ve, it returns a NULL intal. The second parameter is copied to a temporary variable. The 1^{st} parameter is repeatedly multiplied to another variable while the temporary variable keeps decrementing by 1 until the temporary variable goes to 0.