*B.Tech. 3rd Semester Assignment*

**DATA STRUCTURES**

**1.2** Given two big integers a and b of length m and n, multiply them and print their results (Use dynamic memory allocations to create arrays). Note that, big integers cannot be stored even in long int. You need to create an array of m characters to store a m digit integer. You have to write the code the way you did your multiplication in primary school, i.e., multiply, shift, add. Ex: 1234567 x 1111.

**# Algorithm to multiply two big integers, a & b of length m & n, and print their results according to multiplication rule in primary school:**

**Step-1:**

**Step-2:**

**Step-3:**

**Step-4:**

**Step-5:**

**Step-6:**

**Step-7:**

**Step-8:**

**Step-9:**

**Step-10:**

Start

Include the necessary header files.

Declare the functions used in program.

Initialize variables and data structures.

User input for the length of two numbers and check if, they are positive.

Allocate memory for character arrays ‘a’ and ‘b’ to store the first and second numbers of size ‘m+1’ and ‘n+1’ for null terminator.

User input for the first and second number into array ‘a’ and ‘b’.

Check if the lengths of a and b match the specified lengths m and n.

Compare the lengths of a and b to ensure a is the longer number.

Multiplication Analyzation:

**Step-10.1:** Initialize an array ‘result’ of size m + n to store the result.

**Step-10.2:** Define two structures to store the product array and to store an array of product array.

**Step-10.3:** Start iterating through digits of the second number ‘b’ from the right (index ‘n – 1’ to ‘0’).

**Step-10.4:** For each digit in ‘b’, iterate through digits of the first number ‘a’ from the right (index ‘m – 1’ to ‘0’).

**Step-10.5:** Perform primary school style multiplication for each pair of digits and store the products in the product array within the structures.

**Step-10.6:** Print each product row for intermediate products in reverse order, considering carry.

**Step-10.7:** Handle cases when leading zeros need to be omitted and when adding a + sign.

**Step-11:**

**Step-10:**

**Step-11:**

**Step-12:**

**Step-13:**

**Step-14:**

**Step-15:**

**Step-16:**

Initialize an array ‘additionCarry’ to handle carries during addition.

Start iterating through the product array in reverse order from structure.

For each row in the product arrays, iterate through the digits.

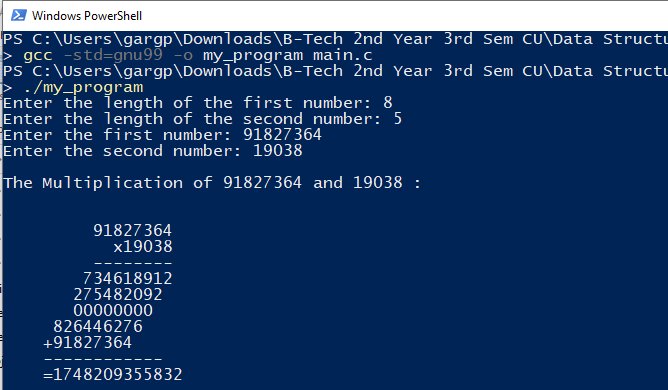
Add the digits, handling the carry from the previous step and updating the addition carry.

Update the result array and carry appropriately.

Add extra decorations for the final output and calculate the correct length of the result and finally print the result in proper format and alignment and write the history and make it save in a file.

Free the memory allocated for additionCarry and result.

End



**# Output:**

**# Flow-Chart:**

**Start**

**Initialize variables and logic for the addition step, perform addition for the final result by adding the partial products and also remove trailing zeros.**

**Prompt the user and read the length of the big integers as input ‘m’ & ‘n’.**

**Print the final result and format the output and also write the history and make it save in a file and also free allocated memory to release resources.**

**Exit with error if negative.**

**Check if m and n are positive integers or not.**

**?**

**End**

**Initialize variables and data structures, perform primary school multiplication using loops and print intermediate products and format output.**

**Swap 'a' and 'b', making a>b.**

**Exit with error if not matched.**

**Allocate memory for arrays ‘a’ and ‘b’.**

**Yes**

**No**

**?**

**Check if n is greater than m.**

**Else continue**

**?**

**Check if the lengths of a and b match m and n.**

**Read the first and second numbers (a and b).**

**Else continue**