Problem Set 1

Handed out: Sunday (15/3/2019): Section 10
Monday (16/3/2019): Section 12

Due: Sunday (22/3/2019): Section 10
Monday (23/3/2019): Section 12

This problem set will introduce you to using 1d and 2d arrays in a broader perspective while solving a problem. This problem set has a single problem which you should save as Your_10_digit_sudent_id.c, e.g. if a student's id is 2013-3-60-23 his code should be saved as 2013-3-60-23.c

Collaboration

You may work with other students and share ideas. But in no circumstances you are allowed to share your code with others. If a case arises as such both the provider and receiver will be treated equally guilty and get a zero.

Before you start: Read the style guide

Read the style guide section from the link provided in the course site.

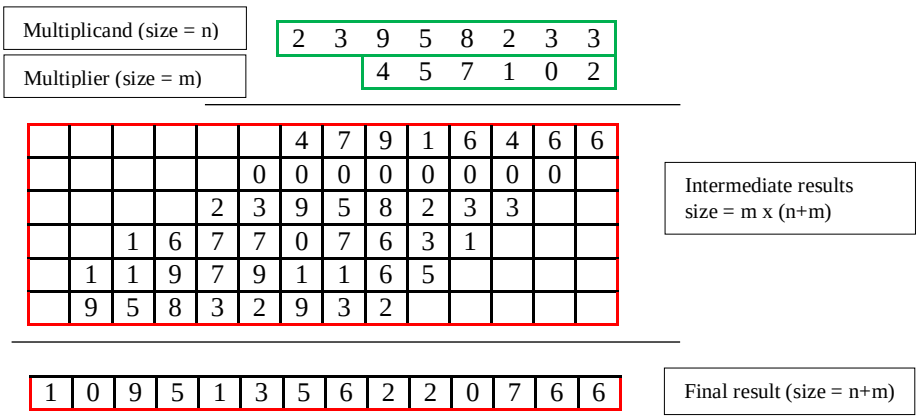
Problem Description

You will be given a multiplicand of `n` and a multiplier of `m` digits where `n, m ≥ 40`. Your task is to find the multiplication of these two numbers. As both `n` and `m` are considerably large you have to write your code accordingly so that they can be taken input in the form of 1D arrays. Similarly, the answer also has to be stored in a 1D array.

The natural way of multiplying numbers taught in the school is known as long multiplication or sometimes known as standard algorithm: multiply the multiplicand by each digit of multiplier and add up all the properly shifted results. We are also going to imply the same rule, only difference is we are going to implement this using arrays.

A. Getting Started (Mark - 5)

The multiplicand and the multiplier will be 1D arrays of size `n` and `m` respectively.
You will also need a 2D array of size `m x (n+m)` to store the intermediate results after multiplying the multiplicand with each digit of multiplier
Finally you will need a 1D array of size `n+m` to store the final result which will be calculated by summing the intermediate results.



B. Finding Intermediate Results (Mark - 20)

Your first task is to calculate the 2D array of intermediate results of size m x (n+m) by multiplying the multiplicand with each digit of multiplier. As the multiplier is of size m, the intermediate result array also contains m rows.

i) Notice that while multiplying the ith (index) digit of the multiplicand with jth (index) digit of multiplier the result is saved in the jth row and ith column of intermediate array.

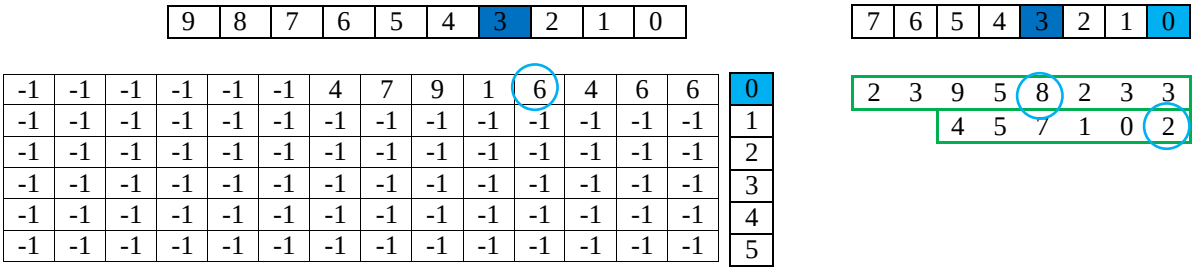


Figure: When you multiply the 3rd index of multiplicand by the 0th index of multiplier the result is saved at 0th row and 3rd column of intermediate array.

ii) Multiplying a multiplicand digit with a multiplier digit may yield a carry or not. For example, when you multiply 2 (0th index of multiplier) with 8 (3rd index of multiplicand) you get 16: then you keep 6 at the 3rd column of 0th row in the intermediate array and pass the carry (=1) to the next digit.
You then find the multiplication of the next index (4th) of multiplicand 5 and 2 which yields a 10 and adds the previous carry 1 to the 10 resulting in 11. Similarly, you put the 1 and pass the 1 as carry to the next digit.

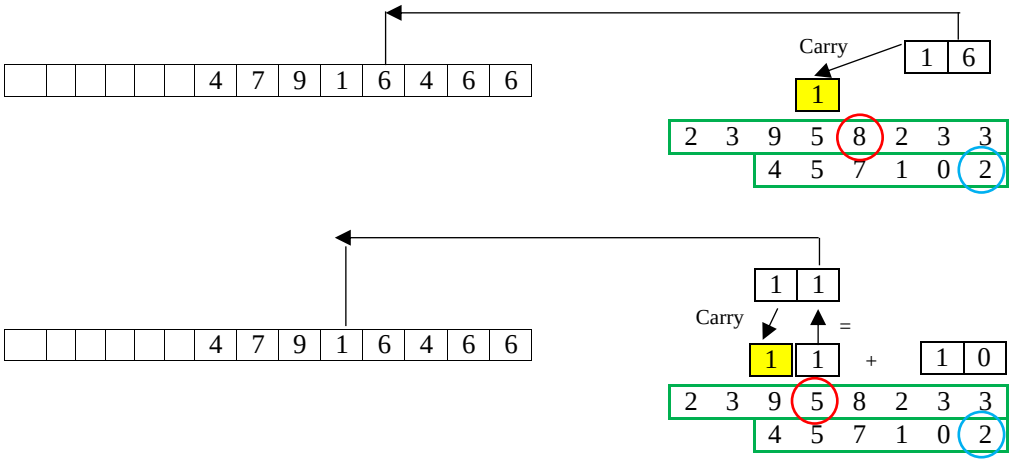


Figure: Handling carries

iii) You also have to be careful of shifting the results appropriate places. For example, when you multiply the 1st index of multiplier with the multiplicands, the resultant answer is saved to the intermediate array after shifting by 1 digit/index. You may use 0s as padding digits.

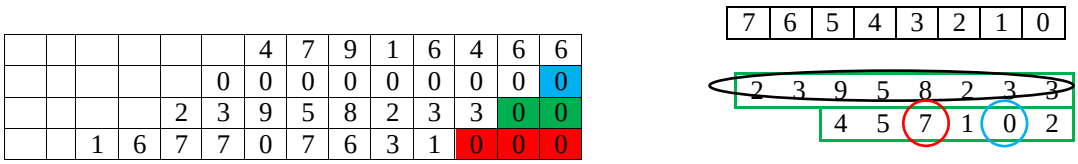


Figure: When you multiply the multiplicand by the 1st and 3rd index of multiplier the result in the intermediate array is padded by 1 and 3 0s respectively.

iv) When you are multiplying the (n-1)th index of a multiplicand with a multiplier index make sure you handle the carry appropriately. For example, when you multiply the last digit (7^{th} index) of multiplicand 2 by 3^{rd} index of multiplier 7 it yields a 16(=14 + 2 from previous carry), so we put a 6 at the [3(padding by 0) + 7(index of multiplicand)] = 10^{th} index and put the carry at 11^{th} index of 3^{rd} row.

Once you are done you will get an intermediate result as below.

-1	-1	-1	-1	-1	-1	4	7	9	1	6	4	6	6
-1	-1	-1	-1	-1	0	0	0	0	0	0	0	0	0
-1	-1	-1	-1	2	3	9	5	8	2	3	3	0	0
-1	-1	1	6	7	7	0	7	6	3	1	0	0	0
-1	1	1	9	7	9	1	1	6	5	0	0	0	0
-1	9	5	8	3	2	9	3	2	0	0	0	0	0

C. Calculating Final Result (Mark - 10)

Now that you are done with your intermediate array, your next task is to calculate the final result. All you got to do is pick each column and find the summation of all non-negative elements. You also have to be careful about handling carries properly. Follow the same instructions as mentioned in Section Finding Intermediate results (II and IV)

-1	-1	-1	-1	-1	-1	4	7	9	1	6	4	6	6
-1	-1	-1	-1	-1	0	0	0	0	0	0	0	0	0
-1	-1	-1	-1	2	3	9	5	8	2	3	3	0	0
-1	-1	1	6	7	7	0	7	6	3	1	0	0	0
-1	1	1	9	7	9	1	1	6	5	0	0	0	0
-1	9	5	8	3	2	9	3	2	0	0	0	0	0

Sample Input-Output

Enter Multiplicand Digits (n): 25
Enter Multiplicands (From n-1 th digit to 0th digit): 6 5 3 0 1 6 1 3 1 9 7 6 3 2 0 1 9 3 1 0 1 2 3 4 5
Enter Multiplier Digits (m): 20
Enter Multiplier (From m-1 th digit to 0^{th} digit): 6 4 4 1 0 1 7 6 5 1 4 8 3 1 0 1 6 3 9 6

Intermediate Results: (From n+m-1 th column to 0^{th} column and 0^{th} row to m-1 th row)
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 3 9 1 8 0 9 6 7 9 1 8 5 7 9 2 1 1 5 8 6 0 7 4 0 7 0
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 5 8 7 7 1 4 5 1 8 7 7 8 6 8 8 1 7 3 7 9 1 1 1 0 5 0
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 9 5 9 0 4 8 3 9 5 9 2 8 9 6 0 5 7 9 3 0 3 7 0 3 5 0 0
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 3 9 1 8 0 9 6 7 9 1 8 5 7 9 2 1 1 5 8 6 0 7 4 0 7 0 0 0 0
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 6 5 3 0 1 6 1 3 1 9 7 6 3 2 0 1 9 3 1 0 1 2 3 4 5 0 0 0 0
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 6 5 3 0 1 6 1 3 1 9 7 6 3 2 0 1 9 3 1 0 1 2 3 4 5 0 0 0 0 0 0
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 9 5 9 0 4 8 3 9 5 9 2 8 9 6 0 5 7 9 3 0 3 7 0 3 5 0 0 0 0 0 0 0
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 5 2 2 4 1 2 9 0 5 5 8 1 0 5 6 1 5 4 4 8 0 9 8 7 6 0 0 0 0 0 0 0 0 0
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 2 6 1 2 0 6 4 5 2 7 9 0 5 2 8 0 7 7 2 4 0 4 9 3 8 0 0 0 0 0 0 0 0 0 0
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 6 5 3 0 1 6 1 3 1 9 7 6 3 2 0 1 9 3 1 0 1 2 3 4 5 0 0 0 0 0 0 0 0 0 0
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 3 2 6 5 0 8 0 6 5 9 8 8 1 6 0 0 9 6 5 5 0 6 1 7 2 5 0 0 0 0 0 0 0 0 0 0 0

-1 -1 -1 -1 -1 -1 -1 3 9 1 8 0 9 6 7 9 1 8 5 7 9 2 1 1 5 8 6 0 7 4 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0
-1 -1 -1 -1 -1 -1 4 5 7 1 1 1 2 9 2 3 8 3 4 2 4 1 3 5 1 7 0 8 6 4 1 5 0 0 0 0 0 0 0 0 0 0 0 0 0
-1 -1 -1 -1 -1 0 6 5 3 0 1 6 1 3 1 9 7 6 3 2 0 1 9 3 1 0 1 2 3 4 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-1 -1 -1 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-1 -1 -1 0 6 5 3 0 1 6 1 3 1 9 7 6 3 2 0 1 9 3 1 0 1 2 3 4 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-1 -1 2 6 1 2 0 6 4 5 2 7 9 0 5 2 8 0 7 7 2 4 0 4 9 3 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-1 2 6 1 2 0 6 4 5 2 7 9 0 5 2 8 0 7 7 2 4 0 4 9 3 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3 9 1 8 0 9 6 7 9 1 8 5 7 9 2 1 1 5 8 6 0 7 4 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Final Result (From n+m-1 th column to 0^{th} column and 0^{th} row to m-1 th row)
4 2 0 6 0 8 8 4 3 2 7 6 2 6 9 7 0 4 1 8 3 5 7 9 0 1 1 8 0 5 1 9 5 4 4 9 5 7 3 4 0 8 6 2 0

Carpe Diem!