**5.Multiplication of big numbers**

Problem Statement:

Write a program to multiply two integers a and b of length m and n respectively.

Take m, n, a and b as input from the user.

Use dynamic memory allocation for storing those big integers.

• Input example :

1. 8

12345678

3

123

1. 27

654154154151454545415415454

32

63516561563156316545145146514654

• Output example :

1. Set Length: 8

Enter num: 12345678

Set Length: 3

Enter num: 123

Multiplication is: 1518518394

1. Set Length: 27

Enter num: 654154154151454545415415454

Set Length: 32

Enter num: 63516561563156316545145146514654

Multiplication is: 41549622603955309777243716069997997007620439937711509062916

Proposed C Code:

/\* ------- main.c ------- \*/

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int main()

{

    int m, n;

    printf("Set Length: ");

    scanf("%d", &m);

    char \*a = (char \*)calloc(m, sizeof(char));

    printf("Enter num: ");

    scanf("%s", a);

    printf("Set Length: ");

    scanf("%d", &n);

    char \*b = (char \*)calloc(n, sizeof(char));

    printf("Enter num: ");

    scanf("%s", b);

    int \*memo = (int \*)calloc((m + n), sizeof(int)); *// initializing an array to store multiplication of chararray b with each element of chararray a*

    int first = 0, second = 0, carry = 0, sum = 0;

    for (int i = m - 1; i >= 0; i--)

    {

        second = 0;

        carry = 0;

        int no1 = a[i] - '0'; *// geting the integer value*

        for (int j = n - 1; j >= 0; j--)

        {

            int no2 = b[j] - '0';

            sum = no1 \* no2 + memo[first + second] + carry; *// addition of multiplication of two elements with previous multiplication shifting by one & the carry*

            carry = sum / 10; *// getting the carry*

            memo[first + second] = sum % 10; *// storing the actual value*

            second++;

        }

        if (carry > 0)

        {

            memo[first + second] += carry;

        }

        first++;

    }

*/\* memo array now have the reverse multiplication\*/*

    int i = m + n - 1;

    while (i > 0 && memo[i] == 0) *// seeing for extra zeros*

    {

        i--;

    }

    char \*ans = (char \*)calloc(i + 1, sizeof(char));

    for (int j = i; j >= 0; j--) *// again reversing the array*

    {

        ans[i - j] = memo[j] + '0';

    }

    printf("Multiplication is: %s", ans);

    return 0;

}

/\* ---------------------- \*/

Conclusion:

The proposed algorithm has a runtime of O(m\*n), where m & n are the length of the corresponding strings.

Limitations and assumptions for this algorithm include:

1. The time complexity of this program is of the order 2 rather than linear.
2. User must enter the strings with declared sizes. Strings which size is greater than their declared size may cause errors.