CSE 2025 DATA STRUCTURES PROJECT#1

Multiplying Large Numbers

Class: Cse 2025 Data Structures

Project Subject/Problem : Multiplying Large Numbers

Name: Mustafa Tunahan

Surname : BAŞ

Student No: 150119055

Submitted To : Birol GENÇYILMAZ

Project Due Date : 05.11.2021 17.00

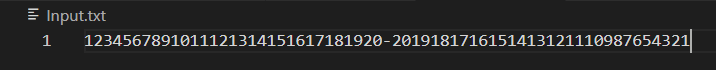
**Problem :** Assume an (theoretically) infinitely large positive natural number specified in decimal number system. How can we multiply it in our computer? We look for a solution that is capable of multiplying such a number. The infinitely large numbers concerning the multiplication (i.e., the multiplicand and the result) should be shown in the original number system presented to your program.

**Implementation in C**

**Contents**

* Input.txt
* Output.txt
* LinkedList.h
* BigNumberMultiplication.c

**Input.txt – Input Format**

****

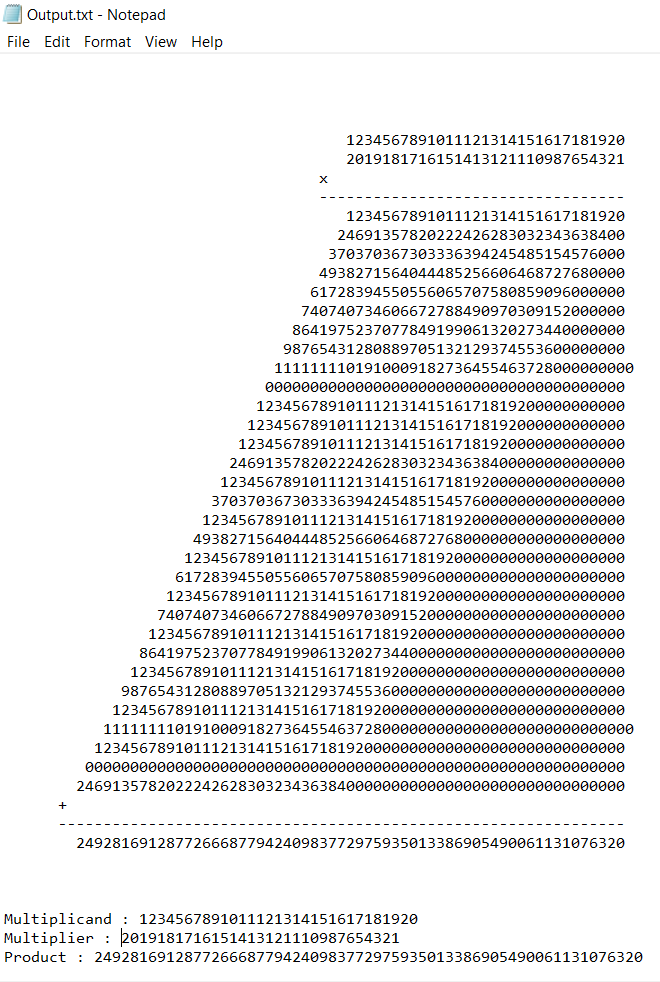
Input.txt consists of two numbers(first is the multiplicand,second is the multiplier) and a non integer character between them,I chose ‘-’ since it makes more sense and looks better but it will work just fine with any other non-integer character as well.

Adding a new line to the input file will probably not break the expected execution of the program since it’s added to my check statements.

Adding multiple lines in this format will not work because the program

currently only reads the first line’s data.

**Output.txt – Output Format**



Output file consists of two parts:

1-Visualisation

2-Basic Results

First part is simply a visualisation of the multiplication operation between the given inputs.

Although it’s not 100% accurate always,since some lines are shifted by one due to the carry in their last digits in the addition.

But it’s mostly accurate because carry in the last digit in an addition is rather a rare event.

Second part simply shows the multiplicand,multiplier and the product.

**LinkedList.h**

This header file consists of functions and structs that help using Linkedlists with basic and common function implementations.

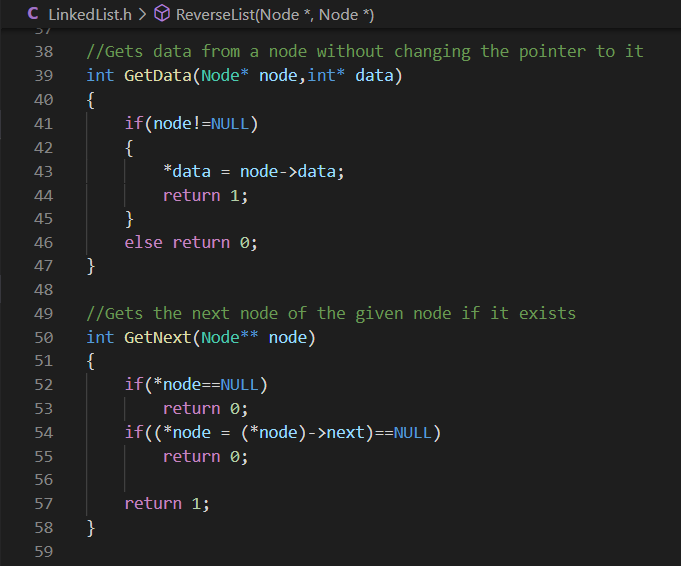
There is a struct definition(a simple linked list node) and 7 function implementations inside it.



Struct Node has two fields,one being the data(int for this particular project),

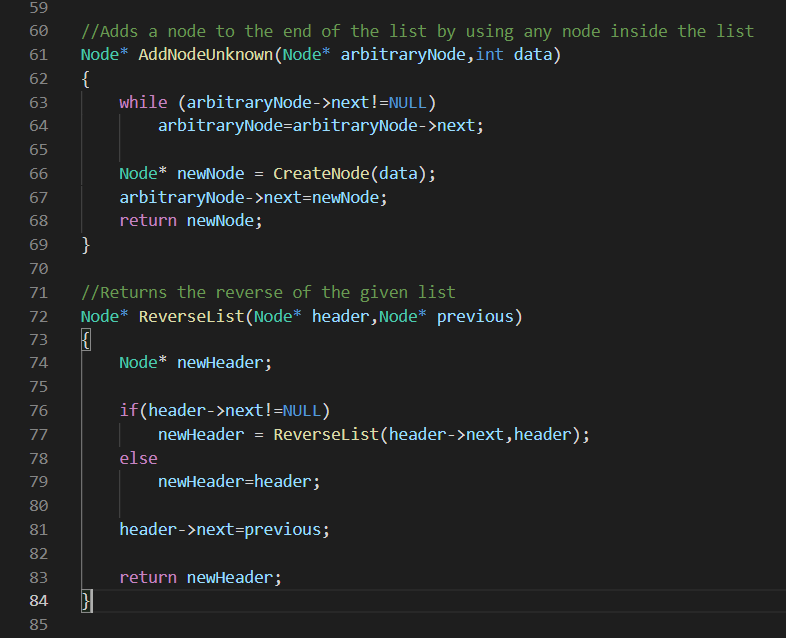
The other one being the next node pointer.

Node\* CreateNode(int data) function simply creates a node with the given data and returns a pointer to it.



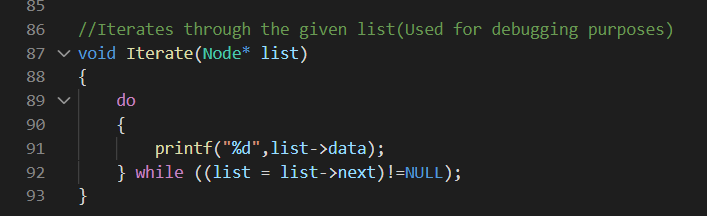
int GetData(Node\* node,int\* data) function gets the data of the given node after the node null-check. If the node is null then 0 is returned, otherwise 1.

int GetNext(Node\*\* node) function gets the next node pointed by the given node if the given node and the next node are not null. It returns a success value of 1 if successful , and 0 if not.



Node\* AddNodeUnknown(Node\* arbitraryNode,int data) function takes an arbitrary node from a list and appends a new node to the end of it.Although I wrote that function, I didn’t use it for the big number multiplication algorithm.

Node\* ReverseList(Node\* header,Node\* previous) function reverses a given list recursively and returns a pointer to the reversed list. Caller should use input NULL to the second parameter of the function unless a new node(previous) will be added to the end of the reversed list



void Iterate(Node\* list) simply iterates through the given list and prints each node to the console, I used it for debugging purposes.

**BigNumberMultiplication.c - Algorithm**

This is the file that the actual algorithm for the big number multiplication is implemented.

It consists a total of 8 functions including the main function.

Text

Description automatically generated

1. **void GetInputs(char\*\* number1,char\*\* number2)**

This function gets the inputs (multiplicand and multiplier) from the input.txt file and puts them into two char arrays.

1-All the input is written to a single char array(line),

2-Then the array is iterated till the non-integer value

3-Non-integer value(at address line+x) will be changed to ‘\0’

4-number2 is given the address of the (line+x+1)

5-number1 is given the address of the line

So I achieved two separate valid char arrays this way.

1. **void WriteOutputs(Node\* number,Node\* number2,Node\* result,Node\*\* nodes,int count)**

This function simply writes the outputs to the output.txt

1. **void WriteList(Node\* node,FILE\* file)**

This function writes the data in the given list to the given file, it’s called from WriteOutputs.

1. **Node\* SplitDigits(char\* number)**

This function is called in the main() function after the inputs are gotten (from the GetInputs function). It creates a list from the char array data(it assumes all of the characters are integers except for string terminator character(NULL))

Iterates through the string till the end of it and gets the digit in each step using int GetDigit(char\*\* number,int\* digit) function,and creates a node each time with the retrieved data from the function.

Returns the header node(list first pointer) After the null character is reached(when GetDigit returns a 0) .

The list returned is the reversed version of the char array.

1. **int GetDigit(char\*\* number,int\* digit**)

This function is called from SplitDigits function to retrieve the digits from the number string.

It converts the current character into an int if the int is not a digit then it returns 0, otherwise 1 is returned and \*digit is assigned. It also alters the pointer address to the next address.

1. **Node\* MultiplyBigNumbers(Node\* first,Node\* second,Node\*\*\* subResults,int\* subResultCount)**

This is the function that multiplies two big numbers(must be reversed) inside two nodes and returns the product of them.

subResults is a return value,it’s an array of Node\* that keeps the data of addition(the lines that are added together after digit by digit multiplication).

subResultCount is also a return value that keeps the length of the subResults array.

1. Second number’s digit count is calculated and stored in an int(say scndCount).
2. An array of Node\* (say nodes )is created dynamically using malloc according to the scndCount(Realize the second number’s length is the count of the numbers that are added together after the multiplication).
3. In a loop,second number’s (multiplier) current digit is calculated using GetData from the LinkedList.h(Realize we have to multiply this digit with every single digit of the first number (multiplicand) to get a line of addition).
4. A Node\* is created using CreateNode from LinkedList.h
5. As many zeros as the index of the addition line are added to the node since we have to shift the each line to the left by the index of it.
6. In another inner loop the current digit of the multiplicand is calculated using GetData function,then res = nm1\*nm2 +carry is calculated.
7. newDigit is calculated using modulus 10 (newDigit = res%10).
8. carry is calculated using division (carry = res/10).
9. New node is added to the end of the current node

10-After the inner loop the complete node(addition line at index x) is added to the end of the nodes array.

11- After the first loop terminates,nodes inside the nodes array are added together using Node\* AddBigNumber(Node\* number1,Node\* number2) function.

12- Return values are set and returned to the caller.

Note: This is a pseudo code and variable names may or may not change in the actual implementation

**7-Node\* AddBigNumbers(Node\* number1,Node\* number2)**

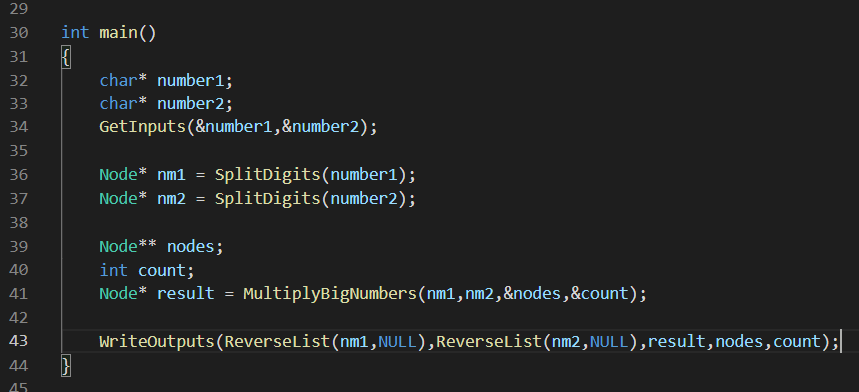
This is the function that adds the nodes and returns the resulting node. It takes two Node\* as parameter(must be reversed).

This is called from MultiplyBigNumbers function to add the resultant lines(numbers) together to get the final result of the multiplication.

1. Result node is created using CreateNode function and another node is created so that it will point to the same address so that we can play with one of the variables without losing the header node.
2. Inside a loop,next data of the numbers are retrieved using GetData function,then two integers(nm1,nm2) are set accordingly(If a number is null after a while then 0 is given as the value to the related number).
3. Result value is calculated (res = nm1+nm2+carry)
4. newDigit is calculated (newDigit = res%10)
5. carry is calculated (carry = res/10)
6. new node created and appended to the result node
7. If both the numbers point to NULL and carry is 0,the loop is terminated
8. The result node is reversed and returned to the caller.

Note: This is a pseudo code and variable names may or may not change in the actual implementation

**8-main()**



1. Inputs are retrieved using GetInputs function
2. Lists are created using SplitDigits function
3. MultiplyBigNumbers is called with the lists
4. WriteOutputs is called and all the data is written to the output.txt(Multiplicand,Multiplier,Result,Multiplication Visualisation)