CSE2025 Data Structures PROJECT #1

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.

Example 1:

```
  530185846923015200254125 3
                           8
x__________________________
 4241486775384121602033002 4
```

Example 2:

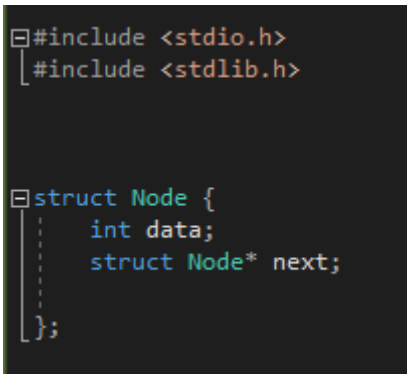
```
  5301
     4
x____
 21204
```


ÖMERCAN SABUN

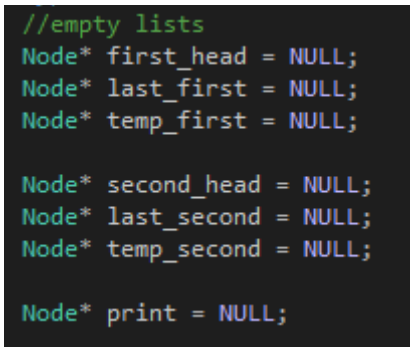
150119555

My Solution

I encountered a lot of bugs in this project. Although I did not specify all of them, I mentioned 2 big bugs. They've been extremely challenging for me.



First, I introduced the linked list node.



Second, i create empty lists because I will use them many times in the future. They are required for the addFirstNode(int data) and addSecondNode(intdata) functions.

```
// multiplicand number as a linked list.
Node addFirstNode(int data) {
    Node* new_node = (Node*)malloc(sizeof(Node)); // allocate node.
    new_node->data = data; // put in the data.
    new_node->next = NULL; // make next of new node as NULL.

    // updating the empty lists(for first) with new_node here.
    if (first_head == NULL) {
        first_head = new_node;
        last_first = first_head;
        temp_first = first_head;
    }
    // traverse till the last node.
    else {
        last_first->next = new_node; // change the next of last node.
        last_first = last_first->next;
    }
    return *new_node;
}
```

I used the empty lists I created earlier here. I created a new node and allocated it and placed the data in it. I set the next one of the node to NULL. If the list named first_head == NULL, I am updating the first_head via the new_node. I also subtract last_first and temp_first from NULL and update with first_head.

If first_head is not null, I update the next element of the first linked list with the node I created. If first_head equals to NULL it goes all the way to the last node and changes the next of the last node.

```
// multiplier number as a linked list.
Node addSecondNode(int data) {
    Node* new_node = (Node*)malloc(sizeof(Node)); // allocate node.
    new_node->data = data; // put in the data.
    new_node->next = NULL; // make next of new node as NULL.

    // updating the empty lists(for second) with new_node here.
    if (second_head == NULL) {
        second_head = new_node;
        last_second = second_head;
        temp_second = second_head;
    }
    // traverse till the last node
    else {
        last_second->next = new_node; // change the next of last node.
        last_second = last_second->next;
    }
    return *new_node;
}
```

The only difference of the function is to create a linked list of multiplier number by using the 3 idle lists I introduced above.

```
Node* createProduct(int data) { //create linked list function for multiply function.

    Node* new_node = (Node*)malloc(sizeof(Node));
    new_node->data = data;
    new_node->next = NULL;

    return new_node;
}
```

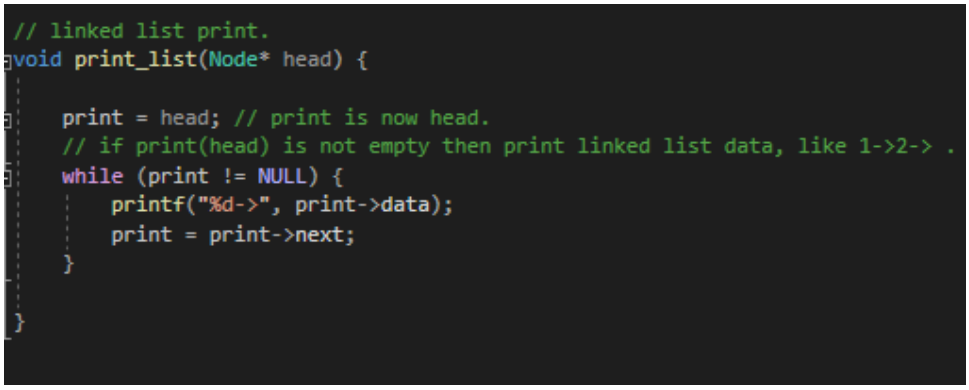
I will use this to create a linked list containing integer data in the function I will create to multiply numbers in the future. malloc frees up memory for the node. It is placing data inside the node data function as integer. The value of the next node connected to the node is set to NULL.

```
// reverse linked list
Node* reverse(Node* head) {
    Node* current = head; //current as head.
    Node* next;
    Node* prev = NULL;
    while (current != NULL) {
        next = current->next; // store next.
        current->next = prev; // reverse current node pointer
        //move pointers one position.
        prev = current;
        current = next;
    }

    return prev;
}
```

Here is the function I created to reverse the linked list. The current node shows head, next and prev are introduced. When current is not null next current->next is equal. I hold the next in my hand and set the next of the current to prev so the pointers change direction and shift one position.

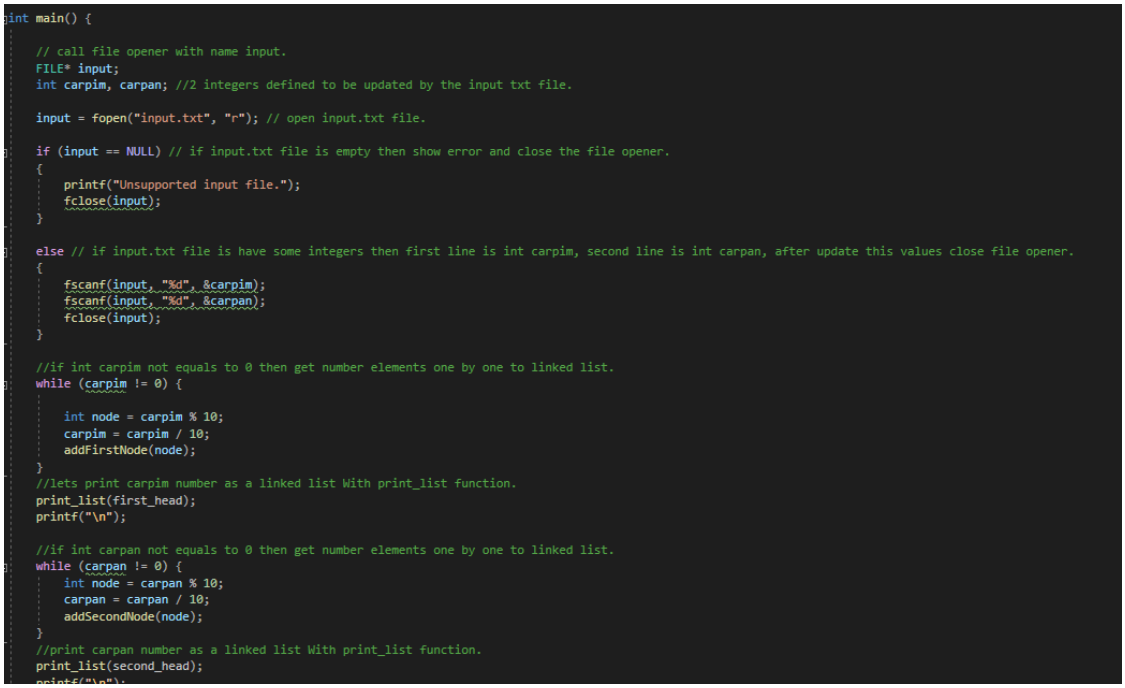
```
Node* multiply(Node* head1, Node* head2) {

    Node* llmultiply;
    llmultiply = createProduct(0);
    Node* result = llmultiply;
    Node* result_check = llmultiply;

    Node* num1 = head1; // we get num1 from head1.
    while (num1) { // there is a loop for the multiplication. ex num1= 152, num2=57. i must multiply all of num1 digits with num2 last digit, then i multiply num1 digits with num2 first digit.


        // this part so important!!! this logic changes everything and this is most confusing part.

        //for ex     if we delete this part
        //  12                                  12
        //  15                                  15
        //x-----      -------->>>              x-----
        //  60                                  60
        // 12        will be like this          12
        //+-----                               +-----
        // 180                                 1260
        //(my first bug was here)
        result_check = result; // result_check updated from result.
        //fixed.
        int carry = 0; // we are multiply digit by digit , for ex num1 = 5 num2 = 3 so num1*num2=15 then we must have carry, carry hold 1 and ->data will be 5.
        Node* num2 = head2; // we get num2 from head2.
        while (num2) {

            result_check->data += carry + (num1->data * num2->data);
            if (result_check->data > 9) {
                carry = result_check->data / 10;
                result_check->data = result_check->data % 10;
            }
            else
                carry = 0;


            if (!result_check->next) { // if result_check dont have next element then set as 0.

                result_check->next = createProduct(0);
            }

            result_check = result_check->next; // go forward

            num2 = num2->next; // we dont want infinite loop so num2 must go forward also.
        }

        if (carry > 0) // we have in the range of 1 to 9 integer, we add carry to result_check->data( next digit by digit multiply) 3*6+(1).
            result_check->data += carry;

        num1 = num1->next; // we dont want infinite loop also num1 must go forward.
        result = result->next; // result go forward.
    }

    llmultiply = reverse(llmultiply); // get linked list reverse// call reverse function//
    if (llmultiply->data == 0) { // if data eqauls to 0 then result will be updated from llmultilpy->next then llmulitply will be erased and recreated and updated from result.
        result = llmultiply->next;
        free(llmultiply);
        llmultiply = result;
    }

    return llmultiply;
}
```

Since digit by digit is multiplied, I carried the remainder to the other side with carry, as the result is not greater than nine. I put number2 in the loop and put number1 in the loop. This is because after multiplying each digit of a number with just one digit of the other number, it will multiply the other digit with a whole number again. The part that I came across was so hard. After 2 hours, I remember that we have to shift once every time we pass a decimal in digit by digit multiplying by drawing on paint. So I got rid of the bugs and reported my examples on the code.



```
// linked list print.
void print_list(Node* head) {

    print = head; // print is now head.
    // if print(head) is not empty then print linked list data, like 1->2-> .
    while (print != NULL) {
        printf("%d->", print->data);
        print = print->next;
    }

}
```

In this part, I print the linked list, if the print node does not return null, it prints the data it has as data->data->data.



```
int main() {

    // call file opener with name input.
    FILE* input;
    int carpim, carpan; //2 integers defined to be updated by the input txt file.

    input = fopen("input.txt", "r"); // open input.txt file.

    if (input == NULL) // if input.txt file is empty then show error and close the file opener.
    {
        printf("Unsupported input file.");
        fclose(input);
    }

    else // if input.txt file is have some integers then first line is int carpim, second line is int carpan, after update this values close file opener.
    {
        fscanf(input, "%d", &carpim);
        fscanf(input, "%d", &carpan);
        fclose(input);
    }

    //if int carpim not equals to 0 then get number elements one by one to linked list.
    while (carpim != 0) {

        int node = carpim % 10;
        carpim = carpim / 10;
        addFirstNode(node);
    }
    //lets print carpim number as a linked list With print_list function.
    print_list(first_head);
    printf("\n");

    //if int carpan not equals to 0 then get number elements one by one to linked list.
    while (carpan != 0) {
        int node = carpan % 10;
        carpan = carpan / 10;
        addSecondNode(node);
    }
    //print carpan number as a linked list With print_list function.
    print_list(second_head);
```

In this part, input.txt file is empty then show error and close the file opener. If input.txt file is have some integers then first line is int carpim, second line is int carpan, after update this values close file opener. (1st line carpim and 2nd line is updating the carpan. // for input.txt)

```
//if int carpim not equals to 0 then get number elements one by one to linked list.
while (carpim != 0) {

    int node = carpim % 10;
    carpim = carpim / 10;
    addFirstNode(node);
}
//lets print carpim number as a linked list With print_list function.
print_list(first_head);
printf("\n");

//if int carpan not equals to 0 then get number elements one by one to linked list.
while (carpan != 0) {
    int node = carpan % 10;
    carpan = carpan / 10;
    addSecondNode(node);
}
//print carpan number as a linked list With print_list function.
print_list(second_head);
printf("\n");

// multiply carpim and carpan linked list then print the result.
// Here I could update the output.txt and then print the output.txt file. I printed the output with this method because it confused me.
print_list(multiply(temp_first, temp_second));
```

If carpim is not equal to 0, I take each digit one by one so that I can add it to the node.After i print first_head so i print first linked list.(carpim number as a linked list). We write the same operations for the carpan and print it as well. After, multiply carpim and carpan linked list then print the result.

```
// call file opener with name output.
FILE* output;
output = fopen("output.txt", "w"); // lets run in write mode.

Node* output_integer; // output_integer linked list.
output_integer = multiply(temp_first, temp_second); // multiply carpim and carpan linked list then update output_integer.
long long int output_number = 0; // initialize output_number.it must be long long int, because we are talking about large large numbers.
while (output_integer) { //  read linked list and transform to integer. this is reversed process. i use in this project int to ll, ll to int.
    output_number += output_integer->data;
    output_number *= 10;
    output_integer = output_integer->next;
}
output_number /= 10; // this is solved my second bug. output.txt show one more 0 at the end of the number. it removes the redundant zero of the number and gives the correct r
fprintf(output, "%lli", output_number); // long long integer output_number going to ---> output.txt.
fclose(output); // close file opener.
```

Finally, I call the file opener in writing mode and convert the linked list result from the multiply function to a long long integer and write it to the output.txt file and close it.

In general, I have shown my examples and the explanation of each line through my code. I hope you will like it sir.