**Project Report**

**Overview**

This project involves creating a program to perform multiplication using linked lists. The program reads two numbers from an input file, stores them in linked lists, multiplies them, and writes the result to an output file. The implementation consists of six functions, including the main function.

**Main Function**

The `main` function manages file operations and coordinates the execution of other functions. It opens the input and output files, reads the multiplicand and multiplier from the input file, performs the multiplication, and writes the result to the output file.

int main() {

FILE \*input\_file, \*output\_file;

input\_file = fopen("C:\\Users\\TAHA\\Desktop\\input.txt", "r");

output\_file = fopen("C:\\Users\\TAHA\\Desktop\\output.txt", "w");

if (input\_file == NULL || output\_file == NULL) {

printf("Error!");

exit(1);

}

char line[150];

Node \*list1, \*list2;

list1 = get\_multiplicand(line, input\_file, output\_file);

list2 = get\_multiplier(line, input\_file, output\_file);

multiplication(&list1, &list2, output\_file);

fclose(input\_file);

fclose(output\_file);

return 0;

}

**Functions**

**1. get\_multiplicand**

This function reads the first number from the input file, creates a linked list to store its digits, and writes the digits to the output file. The linked list is then reversed, and the head of the list is returned.

Node \*get\_multiplicand(char \*line, FILE \*input\_file, FILE \*output\_file) {

fgets(line, 100, input\_file);

Node \*head, \*temp, \*curr;

for (int i = 0; sizeof(line); i++) {

if (line[i] == '\n') {

break;

} else {

curr = (Node \*) malloc(sizeof(Node));

curr->val = line[i] - '0';

fprintf(output\_file, "%d", curr->val);

if (i == 0) {

head = temp = curr;

} else {

temp->next = curr;

temp = curr;

}

}

}

temp->next = NULL;

fprintf(output\_file, "\n");

reverse(&head);

return head;

}

**2. get\_multiplier**

This function reads the second number from the input file and operates similarly to `get\_multiplicand`.

Node \*get\_multiplier(char \*line, FILE \*input\_file, FILE \*output\_file) {

fgets(line, 100, input\_file);

Node \*head, \*temp, \*curr;

for (int i = 0; sizeof(line); i++) {

if (line[i] == '\0') {

break;

} else {

curr = (Node \*) malloc(sizeof(Node));

curr->val = line[i] - '0';

fprintf(output\_file, "%d", curr->val);

if (i == 0) {

head = temp = curr;

} else {

temp->next = curr;

temp = curr;

}

}

}

temp->next = NULL;

fprintf(output\_file, "\n");

reverse(&head);

return head;

}

**3. multiplication**

This function performs the multiplication of two linked lists. It uses several pointers to manage the linked lists and intermediate results. The algorithm involves nested loops to handle digit multiplication and carries. The final result is reversed and printed to the output file.

void multiplication(Node \*\*multiplicand, Node \*\*multiplier, FILE \*output\_file) {

Node \*ptr1 = (\*multiplier);

Node \*ptr2 = (\*multiplicand);

Node \*result, \*tempPtr, \*curr, \*ex\_node, \*tempPtr2, \*ex\_node2;

int digit\_count = 0;

while (ptr1 != NULL) {

int carry = 0, carry2 = 0;

for (int j = 0; j < digit\_count; ++j) {

tempPtr2 = tempPtr2->next;

}

for (int i = 0; ptr2 != NULL; i++) {

if (digit\_count != 0) {

int sum = ptr2->val \* ptr1->val;

tempPtr2->val += sum;

tempPtr2->val += carry2;

carry2 = tempPtr2->val / 10;

tempPtr2->val = tempPtr2->val % 10;

if (ptr2->next == NULL && carry2 != 0) {

ex\_node2 = (Node \*) malloc(sizeof(Node));

tempPtr2->next = ex\_node2;

tempPtr2 = tempPtr2->next;

ex\_node2->val = carry2;

ex\_node2->next = NULL;

} else {

tempPtr2 = tempPtr2->next;

}

} else {

curr = (Node \*) malloc(sizeof(Node));

int sum = ptr2->val \* ptr1->val;

sum += carry;

carry = sum / 10;

sum = sum % 10;

if (i == 0) {

result = curr;

tempPtr = result;

tempPtr2 = result;

} else {

tempPtr->next = curr;

tempPtr = curr;

}

if (ptr2->next == NULL) {

curr->val = sum;

if (carry != 0) {

ex\_node = (Node \*) malloc(sizeof(Node));

tempPtr->next = ex\_node;

tempPtr = ex\_node;

ex\_node->val = carry;

ex\_node->next = NULL;

} else {

curr->next = NULL;

}

} else {

curr->val = sum;

}

}

ptr2 = ptr2->next;

}

digit\_count++;

ptr1 = ptr1->next;

ptr2 = (\*multiplicand);

tempPtr2 = result;

}

reverse(&result);

printList(&result, output\_file);

}

**4. reverse**

This function reverses the linked list.

void reverse(Node \*\*head) {

Node \*previous = NULL;

Node \*current = \*head;

while (current != NULL) {

Node \*next = current->next;

current->next = previous;

previous = current;

current = next;

}

\*head = previous;

}

```

5. \*\*printList\*\*

This function prints the linked list to the output file.

```c

void printList(Node \*\*head, FILE \*output\_file) {

Node \*temp = (\*head);

while (temp != NULL) {

fprintf(output\_file, "%d", temp->val);

temp = temp->next;

}

}

**Conclusion**

The program successfully reads two numbers from an input file, stores them in linked lists, performs their multiplication, and writes the result to an output file. The linked list data structure and file handling are efficiently managed through the described functions. This implementation demonstrates effective use of linked lists for arithmetic operations and file I/O in C.