자료구조 과제(HW4)

전공: 철학과 학년: 3학년 학번: 20180032 이름: 남기동

1. Matrix Transpose

<Pseudo Code>

Global Variable: matrix_pointer hdnode[MAX_SIZE], hdnode_t[MAX_SIZE]

Local Variable: matrix, t_matrix

Function:

1. main function

Make local variable matrix, t_matrix whose datatypes are 'matrix_pointer'

Call mread function and save all information into 'matrix'

Call mtranspose function with argument 'matrix' and get the transposed matrix and save it into 't_matrix'

Call mwrite(t_matrix) to write transposed matrix's information into "output.txt"

Call merase function to erase all nodes in matrix and t_matrix

2. matrix_pointer mread( )

Open "input.txt" file for reading information

Get first line information and it will be the total number of rows, cols, terms

Dynamically allocate node and save the information into it

```
num_heads = bigger one between num_cols and num_rows
If (num_heads is zero) node->right = node;
Else {
        For: i=0 to num_heads //making head nodes
            dynamically allocate temp, hdnode[i] = temp, hdnode[i]->tag = head
            hdnode[i]->right = temp; hdnode[i]->u.next = temp;
        current_row = 0
        last = hdnode[0] //last is the matrix_pointer and the last node in matrix
        For: i=0 to num_terms
            read one line from input.txt to get row, col, value information
            if( row > current_row) then close current row
            dynamically allocate temp and put row, col, value into temp
            last = temp
            hdnode[col]->u.next->down = temp
            hdnode[col]->u.next = temp
        close last row
        close all colum lists
        link all header nodes together
close "input.txt" file and return node
```

3. void mwrite(matrix_pointer node)

```
open “output.txt” for writing purpose
write the information of first line that contains total numbers of row, col, terms
for: i = 0; to node->u.entry.row(total number of row)
    for (temp = head->right; temp != head; temp = temp->right)
        write each information of each line in matrix node
    head = head->u.next; //move to next head
close the “output.txt” file
```

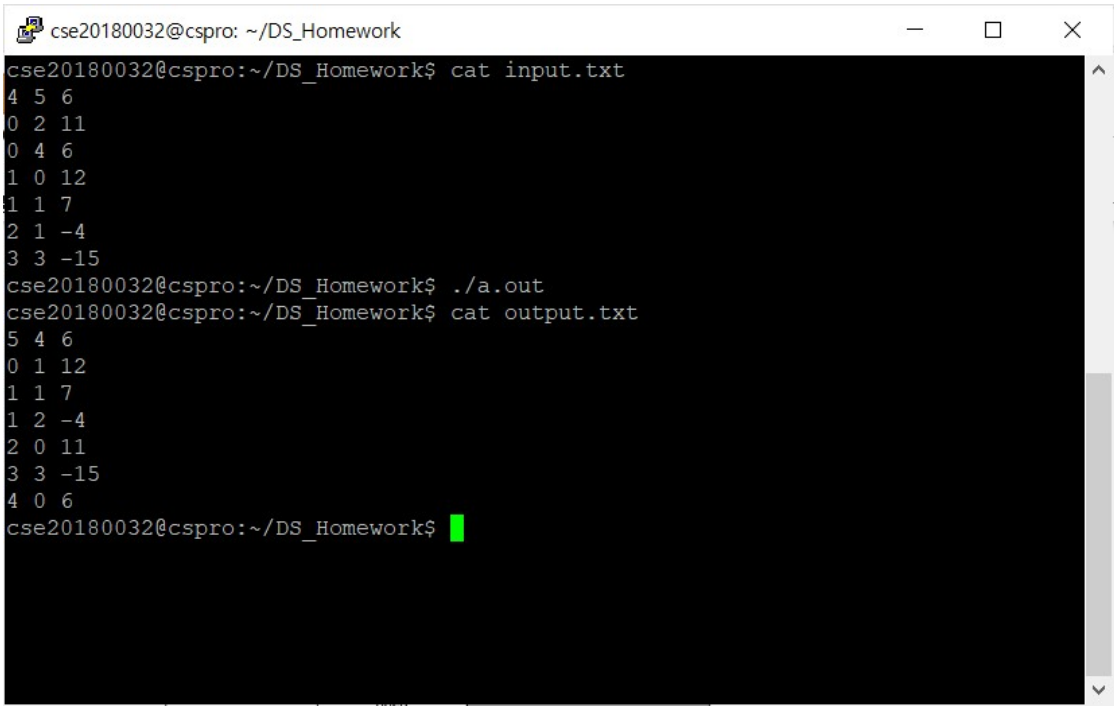
4. void merase(matrix_pointer node)

```
matrix_pointer x, y; matrix_pointer head; int i;
head = node->right;
for: i=0 to node->u.entry.row
    y = head->right;
    while (y != head) { x = y; y = y->right; free(x); } //free all nodes in head
    x = head; head = head->u.next; free(x); //free head itself
//free remaining head nodes
y = head;
while (y != node) { x = y; y = y->u.next; free(x); }
free(node); node = NULL;
```

5. matrix_pointer mtranspose(matrix_pointer matrix)

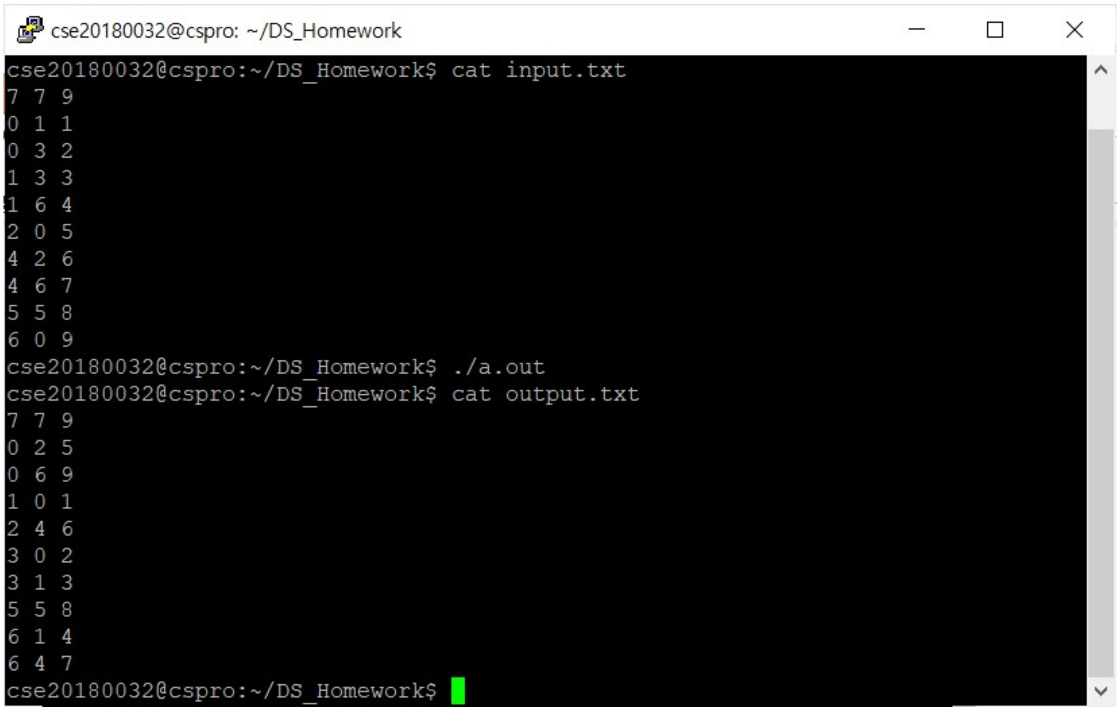
```
Dynamically allocate matrix_pointer 'node'
Put first line information of 'matrix' that contains total number of rows, cols, terms
into node but matrix's row will be node's col and vice versa.
num_heads = bigger one between num_cols and num_rows
If (num_heads is zero) node->right = node;
Else {
        For: i=0 to num_heads //making head nodes
            dynamically allocate temp, hdnode[i] = temp, hdnode[i]->tag = head
            hdnode[i]->right = temp; hdnode[i]->u.next = temp;

        For: j=0 to matrix's cols
                For(ptr = hdnode[i]->down, last hdnode_t[j]; ptr != hdnode[i]; ptr =
                        ptr->down)
                        dynamically allocate temp and put matrix's node's
                        Information with exchange row and col into temp
                        Link into row list
                        Link into colum list
                Last->right = hdnode_t[j] //finish the row list
        Close all column lists
        Link all heard nodes together
} return node
```

<Test Examples>



cse20180032@cspro: ~/DS_Homework

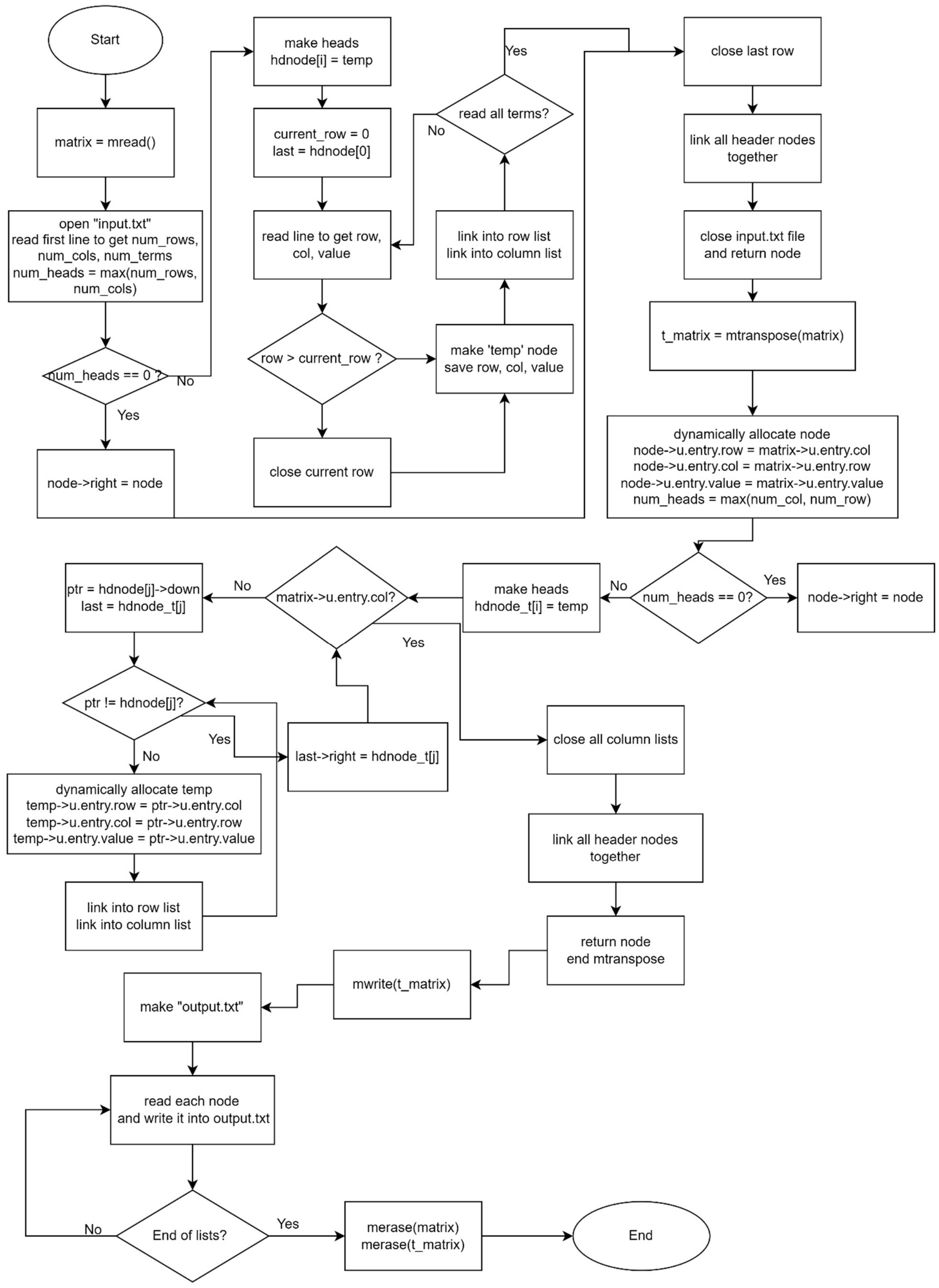
```
cse20180032@cspro:~/DS_Homework$ cat input.txt
4 5 6
0 2 11
0 4 6
1 0 12
1 1 7
2 1 -4
3 3 -15
cse20180032@cspro:~/DS_Homework$ ./a.out
cse20180032@cspro:~/DS_Homework$ cat output.txt
5 4 6
0 1 12
1 1 7
1 2 -4
2 0 11
3 3 -15
4 0 6
cse20180032@cspro:~/DS_Homework$
```



cse20180032@cspro: ~/DS_Homework

```
cse20180032@cspro:~/DS_Homework$ cat input.txt
7 7 9
0 1 1
0 3 2
1 3 3
1 6 4
2 0 5
4 2 6
4 6 7
5 5 8
6 0 9
cse20180032@cspro:~/DS_Homework$ ./a.out
cse20180032@cspro:~/DS_Homework$ cat output.txt
7 7 9
0 2 5
0 6 9
1 0 1
2 4 6
3 0 2
3 1 3
5 5 8
6 1 4
6 4 7
cse20180032@cspro:~/DS_Homework$
```

<Flowchart>


Start
matrix = mread()
open "input.txt"
read first line to get num_rows, num_cols, num_terms
num_heads = max(num_rows, num_cols)
num_heads == 0 ?
No
Yes
node->right = node
make heads
hdnode[i] = temp
current_row = 0
last = hdnode[0]
read line to get row, col, value
row > current_row ?
close current row
make 'temp' node
save row, col, value
link into row list
link into column list
read all terms?
Yes
No
close last row
link all header nodes together
close input.txt file and return node
t_matrix = mtranspose(matrix)
dynamically allocate node
node->u.entry.row = matrix->u.entry.col
node->u.entry.col = matrix->u.entry.row
node->u.entry.value = matrix->u.entry.value
num_heads = max(num_col, num_row)
num_heads == 0?
Yes
node->right = node
No
make heads
hdnode_t[i] = temp
matrix->u.entry.col?
No
Yes
ptr = hdnode[j]->down
last = hdnode_t[j]
ptr != hdnode[j]?
Yes
No
last->right = hdnode_t[j]
dynamically allocate temp
temp->u.entry.row = ptr->u.entry.col
temp->u.entry.col = ptr->u.entry.row
temp->u.entry.value = ptr->u.entry.value
link into row list
link into column list
close all column lists
link all header nodes together
return node
end mtranspose
mwrite(t_matrix)
make "output.txt"
read each node and write it into output.txt
End of lists?
No
Yes
merase(matrix)
merase(t_matrix)
End

2. polynomial multiplication

<Pseudo Code>

Variable: poly_pointer a, poly_pointer b, poly_pointer c(temporary polynomial that contains all combinations of multiplication of a and b without considering same exponent can't come more than once), poly_pointer d(from 'c', integrating coefficients of terms that have same exponent)

Function:

1. main function

```
Poly_pointer a = getpoly_a();
Poly_pointer b = getpoly_b();
Dynamically allocate poly_pointer 'd'
d = pmult(a, b, &num_d);
pwrite(d, num_d);
perase(a), perase(b), perase(c);
```

2. poly_pointer getpoly_a( )

```
Open "a.txt" file for reading information
Read first line to get total number of terms in polynomial
For i=0 to terms
    Read line and get coefficient and exponent
```

```
            Dynamically allocate temp and save coef, expo into it
            Link each node in order that higher exponent come first
Close the file and return start of linked list that contains polynomial info
```

3. poly_pointer getpoly_b( )

```
Open "b.txt" file for reading information
Read first line to get total number of terms in polynomial
For i=0 to terms
            Read line and get coefficient and exponent
            Dynamically allocate temp and save coef, expo into it
            Link each node in order that higher exponent come first
Close the file and return start of linked list that contains polynomial info
```

4. void perase(poly_pointer ptr)

```
For (temp = ptr, trail = ptr;   temp!=NULL ; )
            Temp = temp->link
            Free(trail)
            Trail = temp
```

5. poly_pointer pmul(poly_pointer a, poly_pointer b, int* num_d)

```
For( ptr1 = a ; ptr1 != NULL ; ptr1 = ptr1->link )
```

```
        Coef_a = ptr1->coef; expo_a = ptr1->expon;
        For (ptr2 = b; ptr2 != NULL; ptr2 = ptr2->link )
                Coef_b = ptr2->coef; expo_b = ptr2->expon;
                Expo_c = expo_a + expo_b
                Coef_c = coef_a * coef_b;
                Dynamically allocate temp and save the info into it
                Link each node in order that higher exponent come first
For (ptr = c; ptr!=NULL ; ptr = ptr->link ) //find max expon in poly 'c'
        If(ptr->expon >=max) max = ptr->expon;
For: i=max to 0 (i--)
        For ( ptr1=c; ptr1 !=NULL; ptr1=ptr1->link )
                If(ptr1->expon == i) result += ptr1->coef;
        If( result != 0 )
                Dynamically allocate temp and save info into it
                Link each node in order that higher exponent come first
perase(c) and return d
```

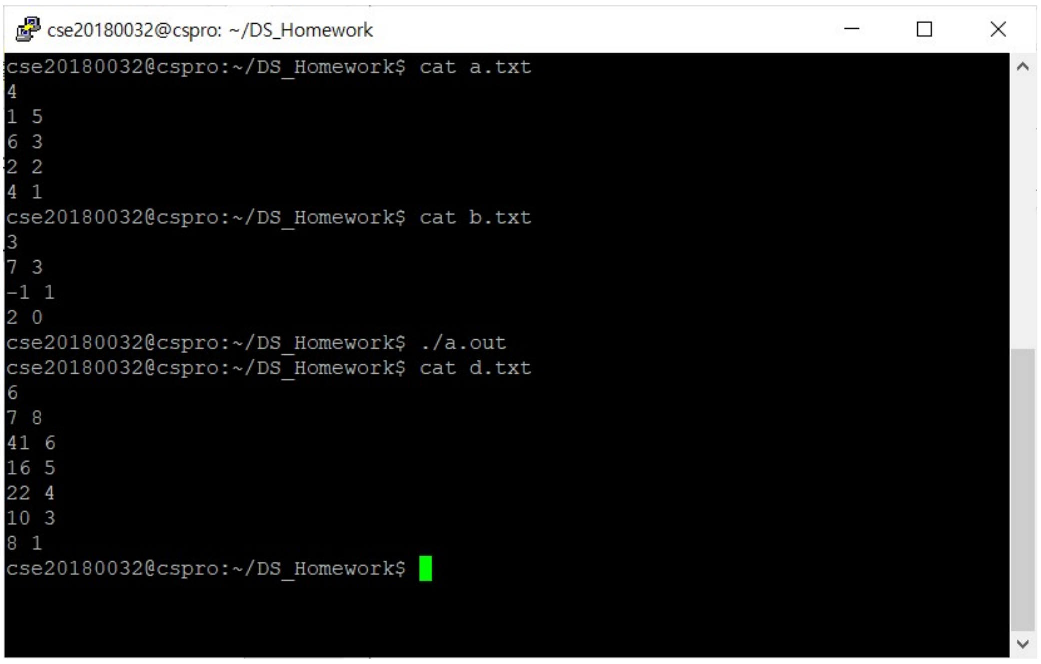
6. void pwrite(poly_pointer d, int num_d)

make "d.txt" to write

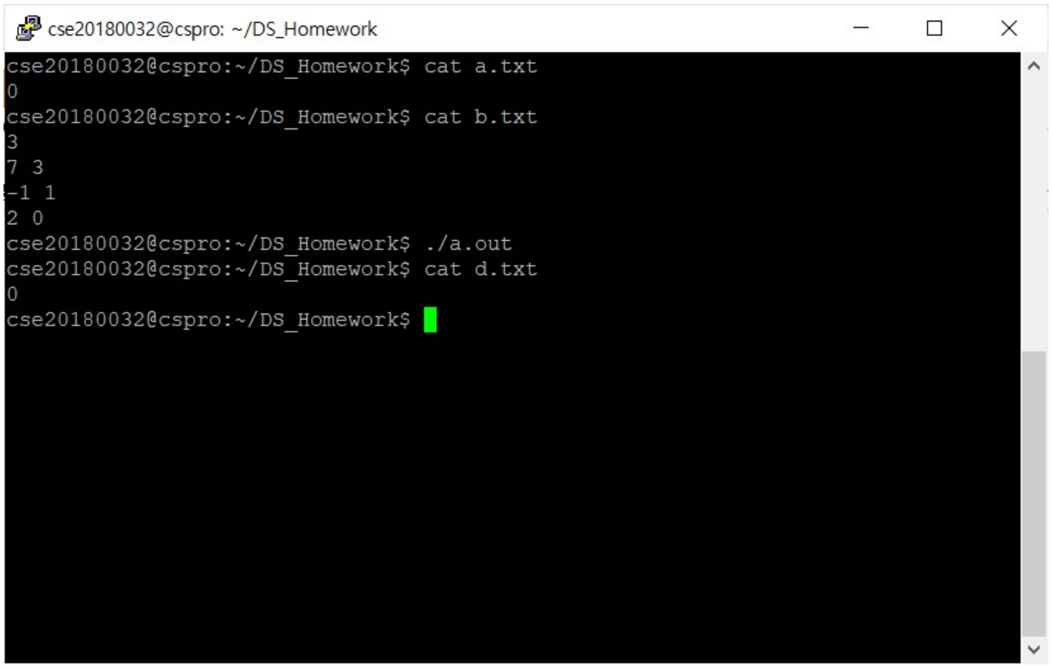
write num_d first which means total number of terms in polynomial

```
for(ptr = d; ptr != NULL; ptr = ptr -> link)
        write coefficients first and exponent next
close the file
```

<Test Examples>

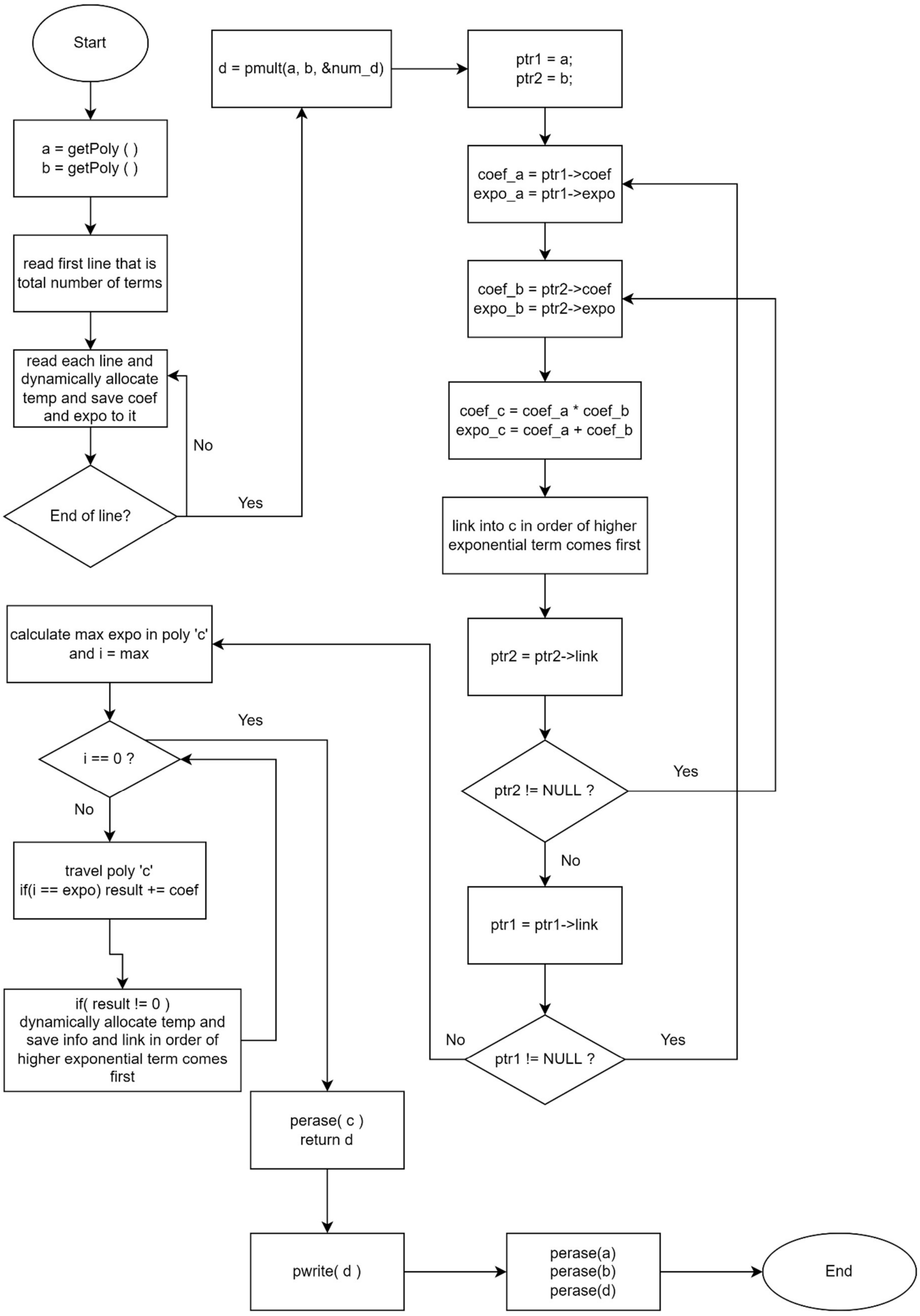

cse20180032@cspro: ~/DS_Homework

```
cse20180032@cspro:~/DS_Homework$ cat a.txt
4
1 5
6 3
2 2
4 1
cse20180032@cspro:~/DS_Homework$ cat b.txt
3
7 3
-1 1
2 0
cse20180032@cspro:~/DS_Homework$ ./a.out
cse20180032@cspro:~/DS_Homework$ cat d.txt
6
7 8
41 6
16 5
22 4
10 3
8 1
cse20180032@cspro:~/DS_Homework$
```


cse20180032@cspro: ~/DS_Homework

```
cse20180032@cspro:~/DS_Homework$ cat a.txt
0
cse20180032@cspro:~/DS_Homework$ cat b.txt
3
7 3
-1 1
2 0
cse20180032@cspro:~/DS_Homework$ ./a.out
cse20180032@cspro:~/DS_Homework$ cat d.txt
0
cse20180032@cspro:~/DS_Homework$
```

<Flowchart>


Start
a = getPoly ( )
b = getPoly ( )
read first line that is total number of terms
read each line and dynamically allocate temp and save coef and expo to it
End of line?
No
Yes
d = pmult(a, b, &num_d)
ptr1 = a;
ptr2 = b;
coef_a = ptr1->coef
expo_a = ptr1->expo
coef_b = ptr2->coef
expo_b = ptr2->expo
coef_c = coef_a * coef_b
expo_c = coef_a + coef_b
link into c in order of higher exponential term comes first
ptr2 = ptr2->link
ptr2 != NULL ?
Yes
No
ptr1 = ptr1->link
ptr1 != NULL ?
Yes
No
calculate max expo in poly 'c' and i = max
i == 0 ?
Yes
No
travel poly 'c'
if(i == expo) result += coef
if(result != 0)
dynamically allocate temp and save info and link in order of higher exponential term comes first
perase(c)
return d
pwrite(d)
perase(a)
perase(b)
perase(d)
End

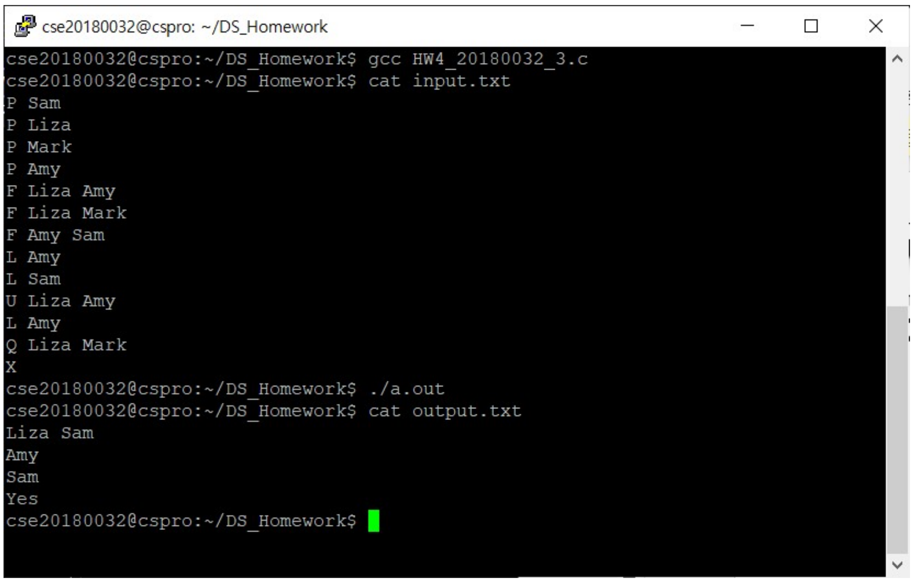
3. Manage SNS friends using equivalence class

<Pseudo Code>

```
Variable: node_pointer stack[MAX_SIZE](store new person), top(index of stack)

Function:

1. void push(char* name)

If( top == MAX_SIZE) return error

temp = new node, strcpy(temp->name, name), temp->link = NULL;

stack[++top] = top;

2. void pop()

If (top == -1) return error

node_pointer ptr, trail;

for(ptr = stack[top]; ptr != NULL; )

        Trail = ptr; ptr = ptr->link; free(trail);

top--;

3. Main function

Fp = fopen "input.txt" and fp2 = fopen "output.txt"

For( ; ; ) {

        Read one character from input.txt

        If(character is '₩r') read one more word

        If(character is '₩n') read one more word
If( character is P or L ) {
    read one word(name1)
    if(character is P)
        if( name1 is in the list ) return error
        if not, push(name1)
    else if(character is L)
        find proper position in stack
        moving with ptr and find friend info of stack and print it
else if( character is F or U or Q)
    read one word(name1) and other word(name2)
    if(character is F)
        if( name1, name2 is same) break;
        if name1, name2 is already friend, break;
        if not,
            temp = new node, temp->name = name2;
            linking temp to the last node of proper stack[i]
            make other temp and put name1 to it
            linking temp to the last node of proper stack[i]
    else if (character is U)
        if( name1, name2 is same) break;
```

```
                    free node that contains name2 in proper stack[i]
                    free node that contains name1 in other proper stack[i]
                else if (character is Q)
                    find stack[i] whose name is same as name1
                    moving by ptr = ptr->link, find whether name2 is in the list
                    if it is in the list, fprintf “yes” into “ouput.txt”
                    if not, fprintf “no” into “output.txt”
        else if (character is X) break
        else return error
}
For (i=top; i>=0; i—) pop() for erase all nodes that were dynamically allocated
Fclose(fp) and fclose(fp2)
```

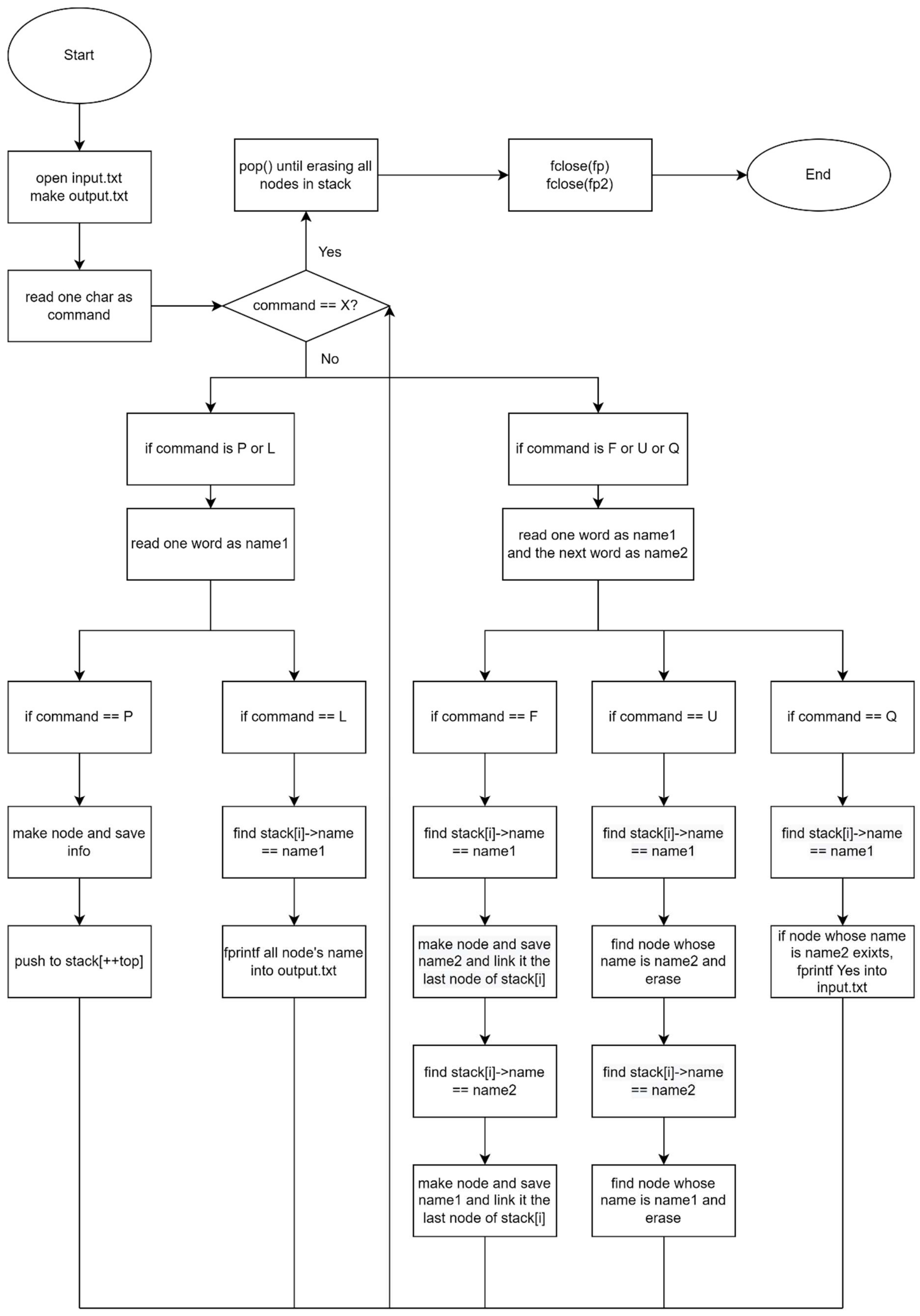
<Test Examples>



cse20180032@cspro: ~/DS_Homework

```
cse20180032@cspro:~/DS_Homework$ gcc HW4_20180032_3.c
cse20180032@cspro:~/DS_Homework$ cat input.txt
P Sam
P Liza
P Mark
P Amy
F Liza Amy
F Liza Mark
F Amy Sam
L Amy
L Sam
U Liza Amy
L Amy
Q Liza Mark
X
cse20180032@cspro:~/DS_Homework$ ./a.out
cse20180032@cspro:~/DS_Homework$ cat output.txt
Liza Sam
Amy
Sam
Yes
cse20180032@cspro:~/DS_Homework$
```

<Flowchart>


Start
open input.txt make output.txt
read one char as command
command == X?
Yes
No
pop() until erasing all nodes in stack
fclose(fp) fclose(fp2)
End
if command is P or L
read one word as name1
if command is F or U or Q
read one word as name1 and the next word as name2
if command == P
make node and save info
push to stack[++top]
if command == L
find stack[i]->name == name1
fprintf all node's name into output.txt
if command == F
find stack[i]->name == name1
make node and save name2 and link it the last node of stack[i]
find stack[i]->name == name2
make node and save name1 and link it the last node of stack[i]
if command == U
find stack[i]->name == name1
find node whose name is name2 and erase
find stack[i]->name == name2
find node whose name is name1 and erase
if command == Q
find stack[i]->name == name1
if node whose name is name2 exixts, fprintf Yes into input.txt