```c
//01
// 2020118008 박보경
// 본인은 이 소스파일을 다른 사람의 소스를 복사하지 않고 직접 작성하였습니다.

#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <stdlib.h>

#define MAX_TERMS 101
#define MAX_COL 6

typedef struct
{
	int row;
	int col;
	int value;
}term;


int avail = 9;


void inputmatrix(term* A, char* fname);
void fastTranspose(term a[], term b[]);
void printmatrix(term* C, int(*ary)[6]);
void inputFile(term* D, char* fname);


int main()
{
	term A[MAX_TERMS]; //a[0]: 행 크기, 열 크기, 0아닌 항의 개수
	term B[MAX_TERMS]; //a의 전치 행렬 b

	char* filename1 = "a.txt";
	char* filename2 = "b.txt";

	int ary[7][6] = { 0 };
	int bry[7][6] = { 0 };


	inputmatrix(A, filename1);
	fastTranspose(A, B);

	printf("A \n");
	printmatrix(A, ary);

	printf("\n");

	printf("B \n");
	printmatrix(B, bry);


	char* filenameB = "b.txt";
	inputFile(B, filenameB);


	return 0;

}

void inputmatrix(term* A, char* fname) //파일로부터 희소행렬을 입력받음
{

	FILE* fp = fopen(fname, "r");

	int row2, col2, value2;

	for (int i = 0; i < avail; i++)
	{
		fscanf_s(fp, "%d %d %d ", &row2, &col2, &value2);
		A[i].row = row2;
		A[i].col = col2;
		A[i].value = value2;
	}
}

void fastTranspose(term a[], term b[])
{
	int rowTerms[MAX_COL], startingPos[MAX_COL];
	int i, j, numCols = a[0].col, numTerms = a[0].value;
	b[0].row=numCols; b[0].col = a[0].row;
	b[0].value = numTerms;

	if (numTerms > 0)
	{
		for (i = 0; i < numCols; i++)
			rowTerms[i] = 0;
		for (i = 1; i <= numTerms; i++)
			rowTerms[a[i].col] ++;

		startingPos[0] = 1;

		for (i = 1; i < numCols; i++)
			startingPos[i] = startingPos[i - 1] + rowTerms[i - 1];

		for (i = 1; i <= numTerms; i++)
		{
			j = startingPos[a[i].col]++;
			b[j].row = a[i].col; b[j].col = a[i].row;
			b[j].value = a[i].value;
		}
	}
}


void printmatrix(term* C, int(*ary)[6])
{
	for (int i = 1; i <= C->value; i++)
	{
		ary[C[i].row][C[i].col] = C[i].value;
	}

	for (int i = 0; i < 7; i++)
	{
		for (int j = 0; j < 6; j++)
		{
			printf("%d ", ary[i][j]);
		}
		printf("\n");
	}
}


void inputFile(term* D, char* fname)
{
	FILE* fp = fopen(fname, "w");
	for (int i = 0; i < avail; i++)
	{
		fprintf(fp, "%d %d %d\n", D[i].row, D[i].col, D[i].value);

	}

}

//02
// 2020118008 박보경
// 본인은 이 소스파일을 다른 사람의 소스를 복사하지 않고 직접 작성하였습니다.

#define _CRT_SECURE_NO_WARNINGS

#include<stdio.h>
#include <string.h>
#include<stdlib.h>
#define MAX_STACK_SIZE 5
#define MAX_NAME_SIZE 10


typedef struct
{
	int id;
	char name[MAX_NAME_SIZE];
}element;

element stack[MAX_STACK_SIZE];

int top = -1;


element stackEmpty();
void stackFull();
void push(element item);
element pop();


int main()
{
	char input[80];
	char* delimiter = " \n";
	char* op = NULL;
```

```c
    element student;
    int cnt;

    printf("<< stack operations where MAX_SIZE is 5>>\n");
    printf("push 1 Jung\n");
    printf("pop\n");
    printf("************************************************\n");

    while (1)
    {
        gets(input);
        op = strtok(input, delimiter);

        if (!strcmp(op, "push"))
        {
            sscanf(input + strlen(op) + 1, "%d%s", &student.id,
student.name);
            push(student);
        }
        else if (!strcmp(op, "pop"))
        {
            element item;
            item = pop();
            if (item.id == -1)
                stackEmpty();

            //if (item.id == 1)
                //exit(EXIT_FAILURE);
        }
        else if (!strcmp(op, "quit"))
            break;
        else
            printf("wrong command!try again!\n");
    }

    return 0;
}


element stackEmpty()
{
    element item;
    item.id = -1;
```

```
        fprintf(stderr, "Stack is empty, cannot delete element");
        return item;
        exit(EXIT_FAILURE);
}

void stackFull()
{
        fprintf(stderr, "Stack is full, cannot add element\n");
        fprintf(stderr, "currnet stack element\n");
        for (int i = MAX_STACK_SIZE-1; i >= 0; i--)
        {
                printf( "%d  %s \n", stack[i].id, stack[i].name);
                pop();
        }

        exit(EXIT_FAILURE);
}

void push(element item) //전역 스택에 아이템 삽입
{
        if (top >= MAX_STACK_SIZE - 1)
                stackFull();
        stack[++top] = item;
}

element pop()
{
        if (top == -1)
                return stackEmpty();
        return stack[top--];
}
```