ASSIGNMENT 12 -62070501064-

Comment Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

#define MAX_COLUMN 30
#define MAX_ROW 30


char command[50];
char params[3][30];
int paramsCount = 0;
//declare the variants to keep in order name and parameter

void ToLower(char *s) { //function to change string to lower character
    int i;
    for(i = 0; s[i]; i++){
        s[i] = tolower(s[i]);
        }
}

void SplitCommand (char cmd[]) { //function to analyse order that user input
    paramsCount = 0; //reset number of parameter

    //delete old parameter
    strcpy(params[0], "");
    strcpy(params[1], "");
    strcpy(params[2], "");

    //split string and keep in order name and parameter
    sscanf(cmd, "%s %s %s %s", command, params[0], params[1], params[2]);

    //count parameter
    if (strcmp(params[0], "") != 0) paramsCount++;
    if (strcmp(params[1], "") != 0) paramsCount++;
    if (strcmp(params[2], "") != 0) paramsCount++;

    ToLower(command); //change into lower for easier to check
}

//function to read matrix file and keep in array
int ReadFile (char *filename, int matrix[][MAX_COLUMN], int *row, int *column)
{
    FILE *file;
    int a;
    //read and check the opening
    if ((file = fopen(filename, "r")) != NULL)
    {
        char line[20][30];
        int lineCount = 0;

        printf("[add] read file %s \n", filename);

        while (!feof(file)) //loop until the end
        {
            if (fgets(line[lineCount], sizeof(line[lineCount]), file))
            {
                lineCount++; //count the line
            }
        }

        //reset number of row and column
        *row = 0;
        *column = 0;

        sscanf(line[0], "%d %d", row, column); //read row line and column line from first

        //set the value to matrix
        int i,j;
        for (i = 0; i < *row; i++) //loop by row
        {
            char *token = strtok(line[i + 1], " "); //split data by space
            for (j = 0; j < *column && token != NULL; j++) // loop by column
            {
                matrix[i][j] = atoi(token); //keep data in matrix
                token = strtok(NULL, " "); // to next data
            }
        }

        fclose(file);
    } else { //show if file can not read
        printf("[error] can't read file.\n");
        return 0;
    }
    return 1;
    }

//function to save matrix in file
void WriteFile (char *filename, int matrix[][MAX_COLUMN], int row, int column)
{
```

Comment Code

```
    int i,j;
    //create file
    FILE *file = fopen(filename, "w");
    if (file != NULL) //check the creating
    {
        //note number of row and column in first line
        fprintf(file, "%d %d\n", row, column);
        //note matrix data
        for (i = 0; i < row; i++) //loop by row
        {
            for (j = 0; j < column; j++) //loop by column
            {
                fprintf(file, "%d ", matrix[i][j]); //note data in row and split by space
            }
            fprintf(file, "\n"); //new line for new row
        }
        fclose(file);
    }
    else { //show if cannot create file
        printf("[error] can't write file.\n");
    }
}

//function to show matrix
void ShowMatrix (int matrix[][MAX_COLUMN], int row, int column)
{
    int i,j;
    printf("MATRIX: %d x %d\n", row, column);
    for (i = 0; i < row; i++)
    {
        for (j = 0; j < column; j++)
        {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");//new line for new row
    }
}

//function to transpose matrix
void Transpose (int matrix[][MAX_COLUMN], int *row, int *column)
{

    int data[*row][MAX_COLUMN]; //create matrix to transpose
    int i,j;
```

```
    for (i = 0; i < *row; i++)
    {
        for (j = 0; j < *column; j++)
        {
            data[i][j] = matrix[i][j]; //transpose and keep in new matrix
        }
    }
    //swap number of row and column
    int _column = * row;
    *row = *column;
    *column = _column;

    //set new matrix in old matrix
    for (i = 0; i < *row; i++)
    {
        for (j = 0; j < *column; j++)
        {
            matrix[i][j] = data[j][i];
        }
    }
}

//function to check correcting of order parameter
int CommandCheck (char *cmd, int *errorParams, int param)
{
    //cmd is order name
    //param is number of wanted parameter
    if (strcmp(command, cmd) == 0)
    {
        if (param == paramsCount)  //number of input parameter must = wanted parameter
        {
            *errorParams = -1; // set to -1 if found order
            return 1;
        }
        else
        {
            char c[50];
            printf("[error] need %d parameter\n", param); //show number of parameter that need
            *errorParams = 1; //set to show error
            return 0;
        }
    }
    else return 0;
}

```

Comment Code

```
//function to check order
void Command (char *cmd)
{

    int errorParams = 0; //declare variant to check order found and error

    find order by CommandCheck
    if (CommandCheck("clear", &errorParams,0)) //clear screen
    {
        system("cls");
    }
    else if (CommandCheck("show", &errorParams, 1))
    {
        //order to show matrix
        int row = 0, column = 0;
        int matrix[30][MAX_COLUMN];
        //check reading matrix and show result
        if (ReadFile(params[0], matrix, &row, &column))
            ShowMatrix(matrix, row, column);
    }
    else if (CommandCheck("transpose", &errorParams, 2))
    {
        //order to transpose matrix
        int row = 0, column = 0;
        int matrix[30][MAX_COLUMN];
        //check reading matrix and continue
        if (ReadFile(params[0], matrix, &row, &column))
        {
           //show matrix before and after tranposing
            ShowMatrix(matrix, row, column);
            Transpose(matrix, &row, &column);
            ShowMatrix(matrix, row, column);
            WriteFile(params[1], matrix, row, column);
        }
    }
    else if (CommandCheck("add", &errorParams, 3))
    {
        //order to summation matrix

        //make 2 matrix
        int aRow = 0, aColumn = 0;
        int bRow = 0, bColumn = 0;
        int aMatrix[MAX_ROW][MAX_COLUMN];
        int bMatrix[MAX_ROW][MAX_COLUMN];
        if (ReadFile(params[0], aMatrix, &aRow, &aColumn)) //must complete reading file1 then continue
        {
            if (ReadFile(params[1], bMatrix, &bRow, &bColumn)) //must complete reading file2 then continue
            {
                //check number of row and column
                if (aRow != bRow || aColumn != bColumn)
                {
                    //if it is not equal then stop process and show warning
                    printf("[error] two matrix not the same size.\n");
                }
                else //if it is equal
                {
                    //make new matrix to be result
                    int cMartix[aRow][MAX_COLUMN];
                    int i,j;
                   //loop every data in new matrix
                    for (i = 0; i < aRow; i++)
                    {
                        for (j = 0; j < aColumn; j++)
                        {
                            //sum matrix A and B then keep in new matrix
                            cMartix[i][j] = aMatrix[i][j] + bMatrix[i][j];
                        }
                    }
                    //show result
                    ShowMatrix(cMartix, aRow, aColumn);
                    //save result
                    WriteFile(params[2], cMartix, aRow, aColumn);
                }
            }
        }
    }
    else if (CommandCheck("remove", &errorParams, 3))
    {
       //prder to minus matrix
        int aRow = 0, aColumn = 0;
        int bRow = 0, bColumn = 0;
        int aMatrix[MAX_ROW][MAX_COLUMN];
        int bMatrix[MAX_ROW][MAX_COLUMN];
        if (ReadFile(params[0], aMatrix, &aRow, &aColumn)) //must complete reading file1 then continue
        {
            if (ReadFile(params[1], bMatrix, &bRow, &bColumn)) //must complete reading file2 then continue
            {
                //check number of row and column
                 if (aRow != bRow || aColumn != bColumn)
                 {
```

Comment Code

```
271                     //if it is not equal then stop process and show warning
272                     printf("[error] two matrix not the same size.\n");
273                 }
274                 else//if it is equal
275                 {
276                    //make new matrix to be result
277                     int cMartix[aRow][MAX_COLUMN];
278                     int i,j;
279                     //minus matrix
280                     for (i = 0; i < aRow; i++)
281                     {
282                         for (j = 0; j < aColumn; j++)
283                         {
284                             //minus and keep in new matrix
285                             cMartix[i][j] = aMatrix[i][j] - bMatrix[i][j];
286                         }
287                     }
288                     //show ans save result
289                     ShowMatrix(cMartix, aRow, aColumn);
290                     WriteFile(params[2], cMartix, aRow, aColumn);
291                 }
292             }
293         }
294     }
295     else if (CommandCheck("multiply", &errorParams, 3))
296     {
297         //order to multiply matrix
298         int aRow = 0, aColumn = 0;
299         int aMatrix[MAX_ROW][MAX_COLUMN];
300         //must complete reading file
301         if (ReadFile(params[0], aMatrix, &aRow, &aColumn))
302         {
303             //make new matirx to be result
304             int cMartix[aRow][MAX_COLUMN];
305             int i,j;
306             printf("%d", atoi(params[1]));
307             for (i = 0; i < aRow; i++)
308             {
309                 for (j = 0; j < aColumn; j++)
310                 {
311                     cMartix[i][j] = aMatrix[i][j] * atoi(params[1]);
312                     //multiply by seconf parameter and keep in new matrix
313                 }
314             }
315             //show and save result
316             ShowMatrix(cMartix, aRow, aColumn);
317             WriteFile(params[2], cMartix, aRow, aColumn);
318         }
319     }
320     else if (CommandCheck("det", &errorParams, 1))
321     {
322         //order to determinant
323         int i,j;
324         int row = 0, column = 0;
325         int matrix[30][MAX_COLUMN];
326         //must complete reading file
327         if (ReadFile(params[0], matrix, &row, &column))
328         {
329             if (row == column) //row and column must be equal
330             {
331                 //show matrix
332                 ShowMatrix(matrix, row, column);
333 
334                 //multiply by formula of determinant
335                 int a = 0, b = 0;
336                 for (j = 0; j < column; j++)
337                 {
338                     int line = matrix[0][j];
339                     for (i = 1; i < row; i++)
340                     {
341                         line = line * matrix[i][(j + i)%(column)];
342                     }
343                     a += line;
344                 }
345                 for (j = 0; j < column; j++)
346                 {
347                     int line = matrix[row-1][j];
348                     for (i = row - 2; i >= 0; i--)
349                     {
350                         line = line * matrix[i][(j + ((row-1) - i))%(column)];
351                     }
352                     b += line;
353                 }
354                 //show result
355                 printf("Det is %d \n", a - b);
356             }
357             else
358             {
359                 printf("[error] matrix must be same size.\n");
360                 //show if row and column are not equal
```

Comment Code

```
361             }
362         }
363     }
364
365     //if erroeParams is 0 : can not find order that user input
366     if (errorParams == 0)
367         printf("[error] commmand not found.\n");
368 }
369
370 int main ()
371 {
372     do
373     {
374         //get new order
375         rewind(stdin);
376         printf("==>> ");
377         char cmd[100];
378         gets(cmd);
379         SplitCommand(cmd);// analyse order
380
381         //if it is not end or exit then continue
382         if (strcmp(command, "end") != 0 && strcmp(command, "exit") != 0)
383             Command(command);
384
385     } while(strcmp(command, "end") != 0 && strcmp(command, "exit") != 0);
386     //if it is end or exit then end program
387     return 0;
388 }
```

Test Case

1.คำสั่งผิด

```
==>> hello world
[error] commmand not found.
```

2.พารามิเตอร์ไม่ครบ หรือเกิน

```
==>> add ma.txt
[error] need 3 parameter
```

3.คำสั่งทำงานไม่ได้ เช่น ขนาดพารามิเตอร์ที่นำมาทำงานไม่สอดคล้องกัน

```
==>> add ma.txt mc.txt md.txt
[add] read file ma.txt
[add] read file mc.txt
[error] two matrix not the same size.
```

4.ทำงานได้ถูกต้อง

```
==>> show ma.txt
[add] read file ma.txt
MATRIX: 4 x 3
1 2 3
4 5 6
7 8 9
10 11 12
==>> show mb.txt
[add] read file mb.txt
MATRIX: 4 x 3
1 4 7
2 5 8
3 6 9
10 11 12
```

คำสั่ง show

```
==>> transpose mb.txt md.txt
[add] read file mb.txt
MATRIX: 3 x 4
1 4 7 10
2 5 8 11
3 6 9 12
MATRIX: 4 x 3
1 2 3
4 5 6
7 8 9
10 11 12
```

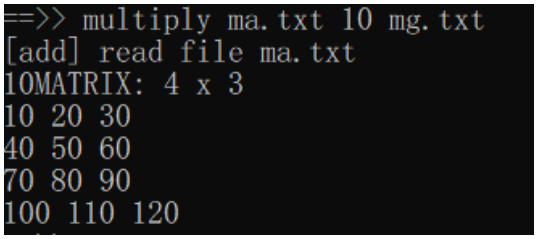
คำสั่ง transpose

```
==>> add ma.txt md.txt me.txt
[add] read file ma.txt
[add] read file md.txt
MATRIX: 4 x 3
2 4 6
8 10 12
14 16 18
20 22 24
```

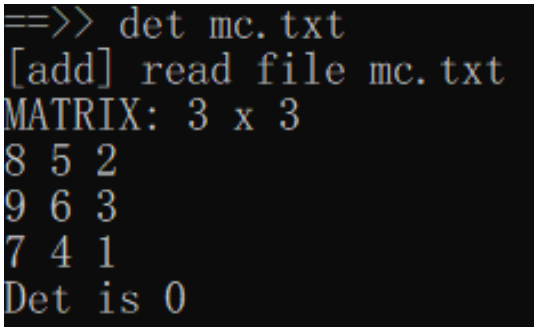
คำสั่ง add /บวก

```
==>> remove me.txt ma.txt mf.txt
[add] read file me.txt
[add] read file ma.txt
MATRIX: 4 x 3
1 2 3
4 5 6
7 8 9
10 11 12
```

คำสั่ง remove /ลบ



คำสั่ง multiply /คูณ



คำสั่ง det

- ปัญหาที่พบในการทำ ASSIGNMENT

สับสนคำสั่งอยู่บ้าง ต้องทำความเข้าใจหลายครั้ง

ลืมเรื่อง matrix

- Self-Assessment : 3 เข้าใจแต่มีปัญหาบางช่วงยังต้องขอความช่วยเหลือ