HACETTEPE
ÜNİVERSİTESİ

NAME: SADIK CAN ACAR

ID: 21626843

PROGRAMMING ASSIGNMENT 1

1)Problem Definition

In this experiment, our problem is to read the matrices and vectors and make operations according to the specified commands.

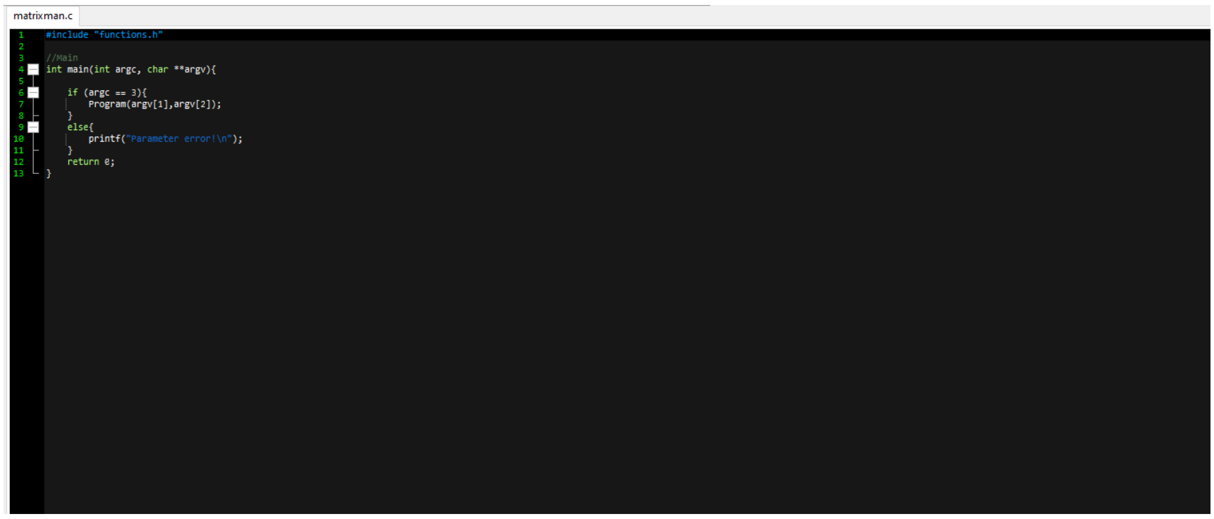
2)Methods and Solution

I could only read the files and create matrices,vectors. Also I can’t write the output to the output file. So I’d appreciate it if you’d run my code like this:

./matrixman [folder_name] commands.txt

3)Functions

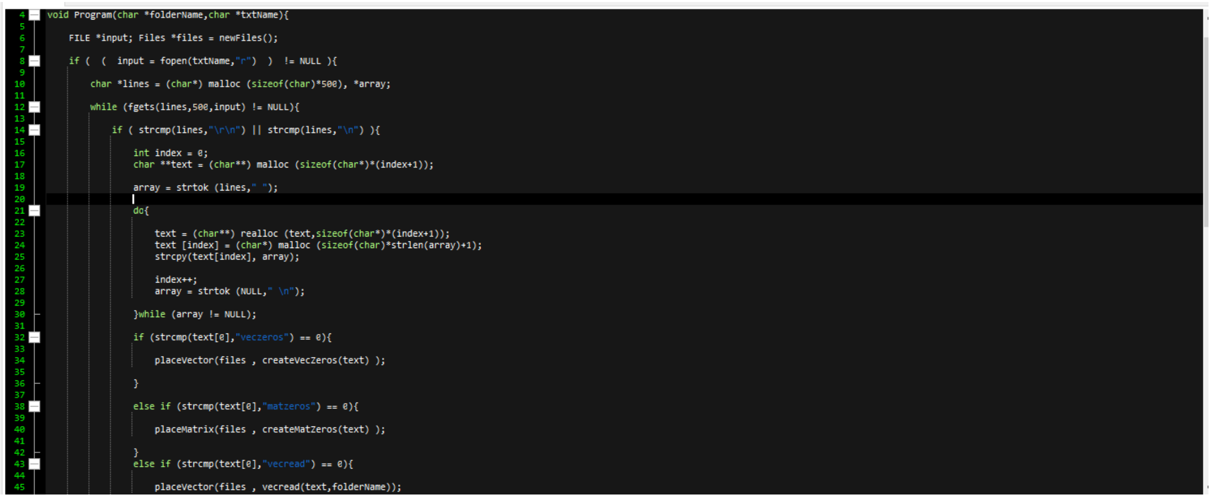
3.a) Main function takes parameters and sends to program function.



matrixman.c

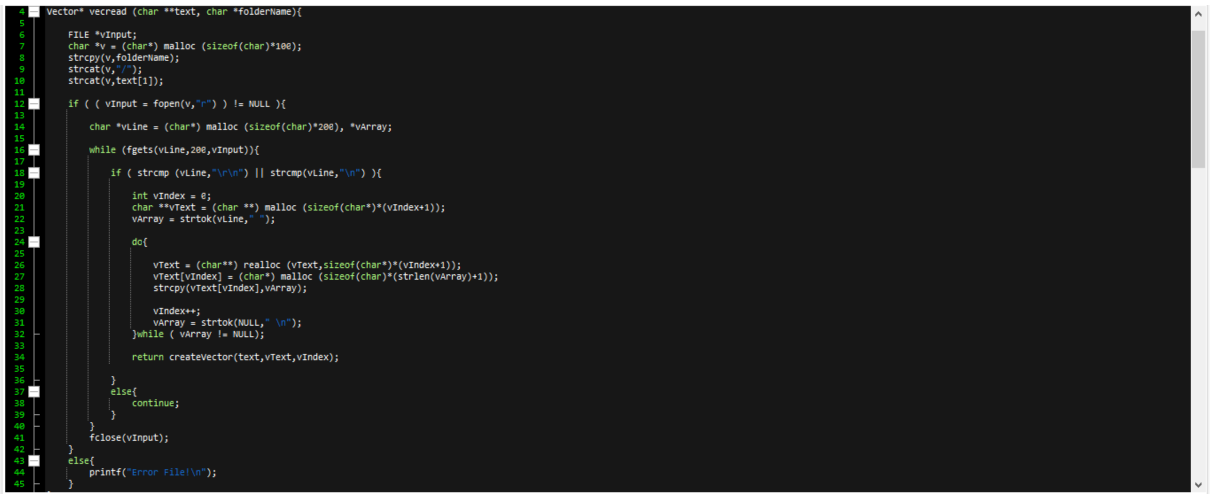
```c
#include "functions.h"

//Main
int main(int argc, char **argv){

    if (argc == 3){
        Program(argv[1],argv[2]);
    }
    else{
        printf("Parameter error!\n");
    }
    return 0;
}
```

3.b) Program function reads commands files and sends to other functions commands and contents.



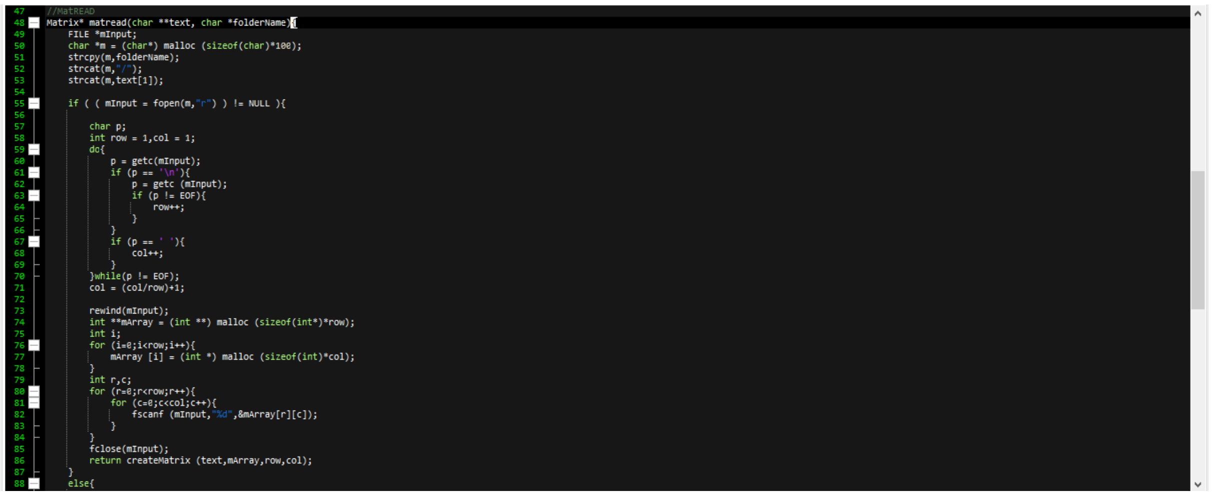
```c
void Program(char *folderName,char *txtName){

    FILE *input; Files *files = newFiles();

    if (  (  input = fopen(txtName,"r")  )  != NULL ){

        char *lines = (char*) malloc (sizeof(char)*500), *array;

        while (fgets(lines,500,input) != NULL){

            if ( strcmp(lines,"\r\n") || strcmp(lines,"\n") ){

                int index = 0;
                char **text = (char**) malloc (sizeof(char*)*(index+1));

                array = strtok (lines," ");

                do{

                    text = (char**) realloc (text,sizeof(char*)*(index+1));
                    text [index] = (char*) malloc (sizeof(char)*strlen(array)+1);
                    strcpy(text[index], array);

                    index++;
                    array = strtok (NULL," \n");

                }while (array != NULL);

                if (strcmp(text[0],"veczeros") == 0){

                    placeVector(files , createVecZeros(text) );

                }

                else if (strcmp(text[0],"matzeros") == 0){

                    placeMatrix(files , createMatZeros(text) );

                }
                else if (strcmp(text[0],"vecread") == 0){

                    placeVector(files , vecread(text,folderName));
```

3.c) Vecread function in commands.c reads vector files and send to createVector function.



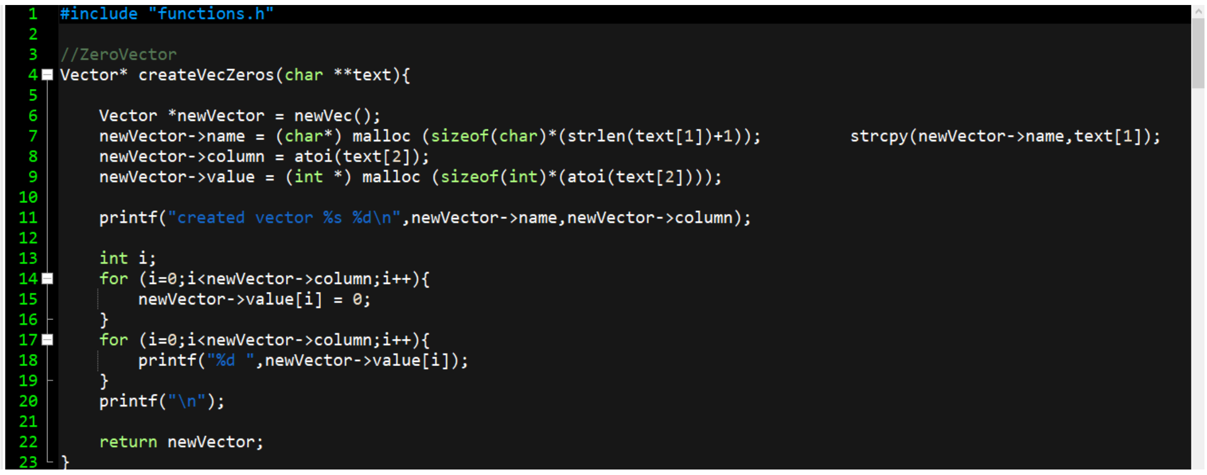
```c
Vector* vecread (char **text, char *folderName){

    FILE *vInput;
    char *v = (char*) malloc (sizeof(char)*100);
    strcpy(v,folderName);
    strcat(v,"/");
    strcat(v,text[1]);

    if ( ( vInput = fopen(v,"r") ) != NULL ){

        char *vLine = (char*) malloc (sizeof(char)*200), *vArray;

        while (fgets(vLine,200,vInput)){

            if ( strcmp (vLine,"\r\n") || strcmp(vLine,"\n") ){

                int vIndex = 0;
                char **vText = (char **) malloc (sizeof(char*)*(vIndex+1));
                vArray = strtok(vLine," ");

                do{

                    vText = (char**) realloc (vText,sizeof(char*)*(vIndex+1));
                    vText[vIndex] = (char*) malloc (sizeof(char)*(strlen(vArray)+1));
                    strcpy(vText[vIndex],vArray);

                    vIndex++;
                    vArray = strtok(NULL," \n");
                }while ( vArray != NULL);

                return createVector(text,vText,vIndex);

            }
            else{
                continue;
            }
        }
        fclose(vInput);
    }
    else{
        printf("Error File!\n");
    }
```

3.d) Matread function in commands.c read matrix files and send to createMatrix function.



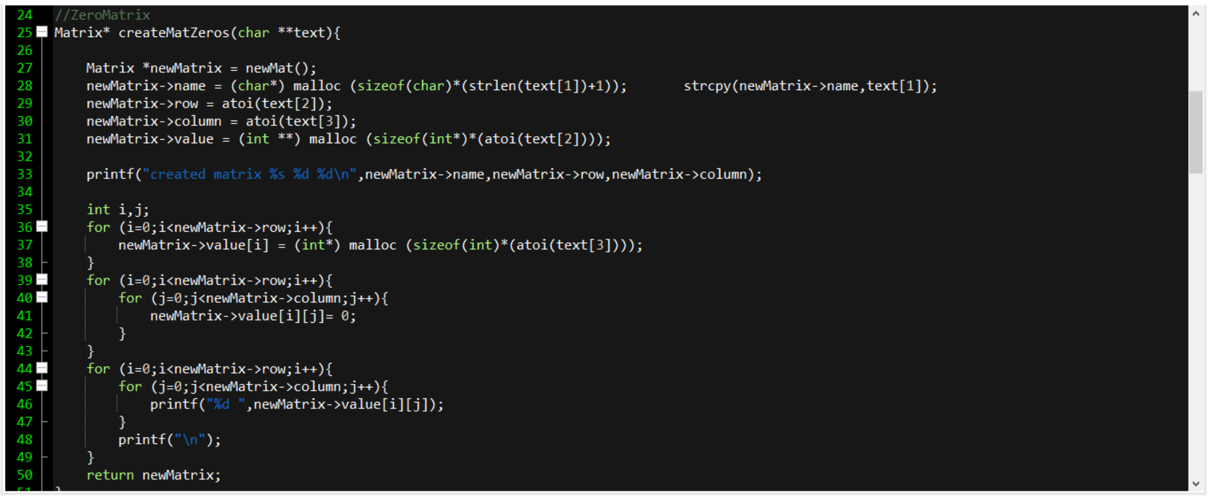
```c
//MatREAD
Matrix* matread(char **text, char *folderName){
    FILE *mInput;
    char *m = (char*) malloc (sizeof(char)*100);
    strcpy(m,folderName);
    strcat(m,"/");
    strcat(m,text[1]);

    if ( ( mInput = fopen(m,"r") ) != NULL ){

        char p;
        int row = 1,col = 1;
        do{
            p = getc(mInput);
            if (p == '\n'){
                p = getc (mInput);
                if (p != EOF){
                    row++;
                }
            }
            if (p == ' '){
                col++;
            }
        }while(p != EOF);
        col = (col/row)+1;

        rewind(mInput);
        int **mArray = (int **) malloc (sizeof(int*)*row);
        int i;
        for (i=0;i<row;i++){
            mArray [i] = (int *) malloc (sizeof(int)*col);
        }
        int r,c;
        for (r=0;r<row;r++){
            for (c=0;c<col;c++){
                fscanf (mInput,"%d",&mArray[r][c]);
            }
        }
        fclose(mInput);
        return createMatrix (text,mArray,row,col);
    }
    else{
```

3.e) CreateVecZeros function creates a zero vector with given length and name.



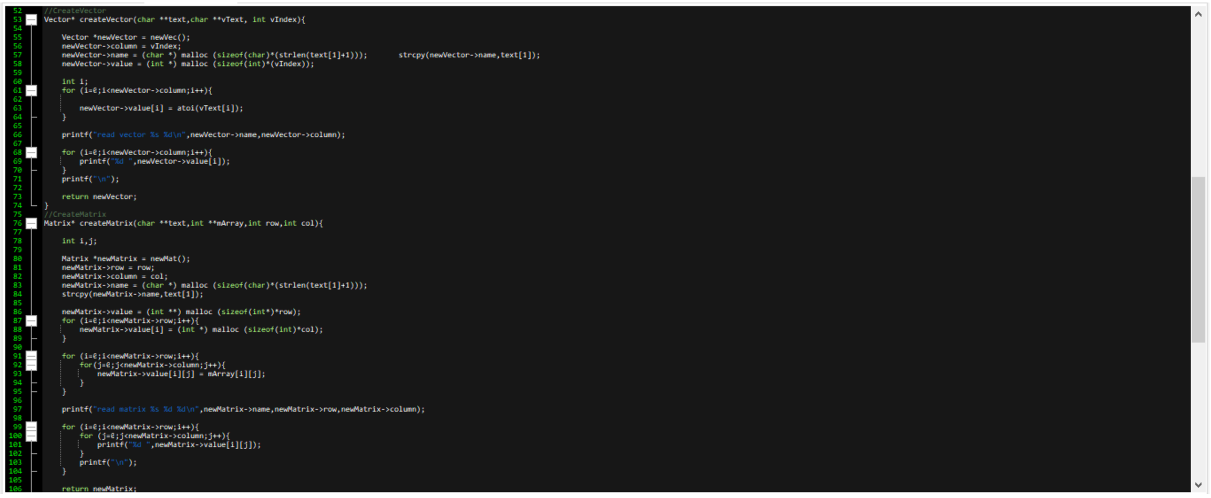
```c
#include "functions.h"

//ZeroVector
Vector* createVecZeros(char **text){

    Vector *newVector = newVec();
    newVector->name = (char*) malloc (sizeof(char)*(strlen(text[1])+1));         strcpy(newVector->name,text[1]);
    newVector->column = atoi(text[2]);
    newVector->value = (int *) malloc (sizeof(int)*(atoi(text[2])));

    printf("created vector %s %d\n",newVector->name,newVector->column);

    int i;
    for (i=0;i<newVector->column;i++){
        newVector->value[i] = 0;
    }
    for (i=0;i<newVector->column;i++){
        printf("%d ",newVector->value[i]);
    }
    printf("\n");

    return newVector;
}
```

3.f) CreateMatZeros creates a zero matrix with given row,column and name.



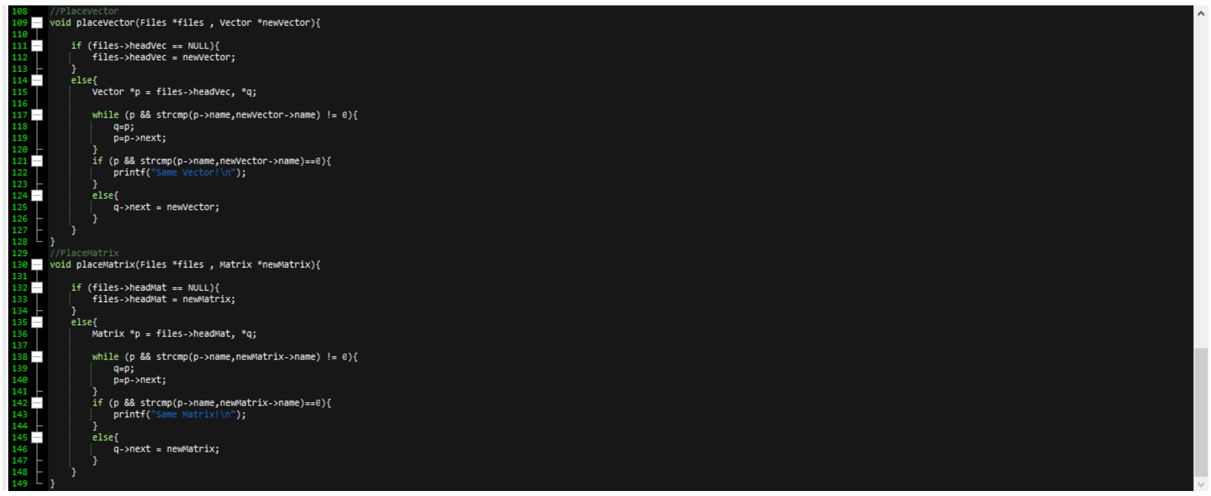
```c
//ZeroMatrix
Matrix* createMatZeros(char **text){

    Matrix *newMatrix = newMat();
    newMatrix->name = (char*) malloc (sizeof(char)*(strlen(text[1])+1));       strcpy(newMatrix->name,text[1]);
    newMatrix->row = atoi(text[2]);
    newMatrix->column = atoi(text[3]);
    newMatrix->value = (int **) malloc (sizeof(int*)*(atoi(text[2])));

    printf("created matrix %s %d %d\n",newMatrix->name,newMatrix->row,newMatrix->column);

    int i,j;
    for (i=0;i<newMatrix->row;i++){
        newMatrix->value[i] = (int*) malloc (sizeof(int)*(atoi(text[3])));
    }
    for (i=0;i<newMatrix->row;i++){
        for (j=0;j<newMatrix->column;j++){
            newMatrix->value[i][j]= 0;
        }
    }
    for (i=0;i<newMatrix->row;i++){
        for (j=0;j<newMatrix->column;j++){
            printf("%d ",newMatrix->value[i][j]);
        }
        printf("\n");
    }
    return newMatrix;
```

3.g) CreateMatrix and CreateVector create matrices and vectors that readed from files.



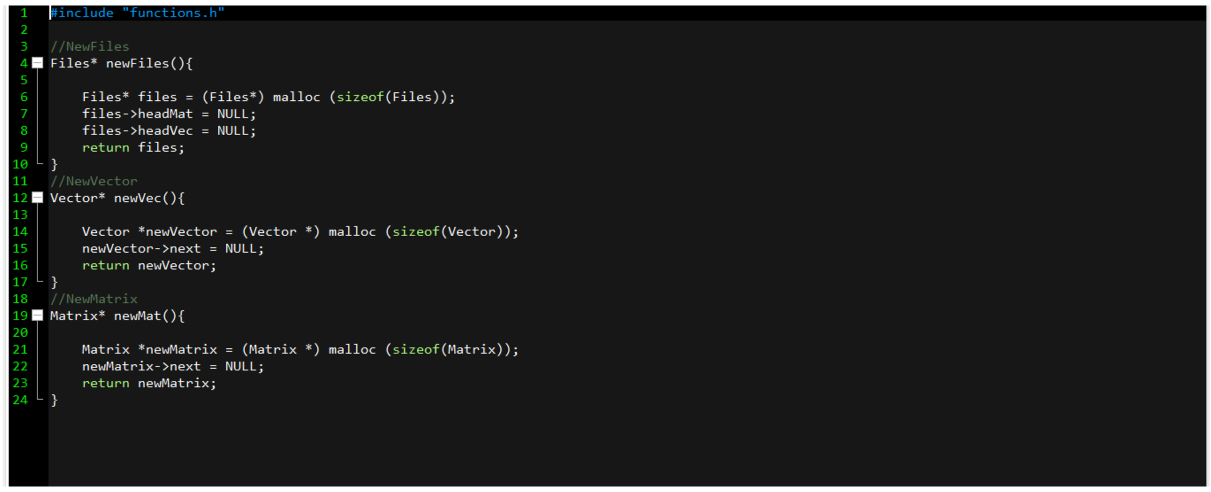
```c
//CreateVector
Vector* createVector(char **text,char **vText, int vIndex){

    Vector *newVector = newVec();
    newVector->column = vIndex;
    newVector->name = (char *) malloc (sizeof(char)*(strlen(text[1]+1)));       strcpy(newVector->name,text[1]);
    newVector->value = (int *) malloc (sizeof(int)*(vIndex));

    int i;
    for (i=0;i<newVector->column;i++){

        newVector->value[i] = atoi(vText[i]);
    }

    printf("read vector %s %d\n",newVector->name,newVector->column);

    for (i=0;i<newVector->column;i++){
        printf("%d ",newVector->value[i]);
    }
    printf("\n");

    return newVector;
}
//CreateMatrix
Matrix* createMatrix(char **text,int **mArray,int row,int col){

    int i,j;

    Matrix *newMatrix = newMat();
    newMatrix->row = row;
    newMatrix->column = col;
    newMatrix->name = (char *) malloc (sizeof(char)*(strlen(text[1]+1)));
    strcpy(newMatrix->name,text[1]);

    newMatrix->value = (int **) malloc (sizeof(int*)*row);
    for (i=0;i<newMatrix->row;i++){
        newMatrix->value[i] = (int *) malloc (sizeof(int)*col);
    }

    for (i=0;i<newMatrix->row;i++){
        for(j=0;j<newMatrix->column;j++){
            newMatrix->value[i][j] = mArray[i][j];
        }
    }

    printf("read matrix %s %d %d\n",newMatrix->name,newMatrix->row,newMatrix->column);

    for (i=0;i<newMatrix->row;i++){
        for (j=0;j<newMatrix->column;j++){
            printf("%d ",newMatrix->value[i][j]);
        }
        printf("\n");
    }

    return newMatrix;
```

3.h) PlaceVector and placeMatrix place matrices and vectors to the structure.



```c
//PlaceVector
void placeVector(Files *files , Vector *newVector){

    if (files->headVec == NULL){
        files->headVec = newVector;
    }
    else{
        Vector *p = files->headVec, *q;

        while (p && strcmp(p->name,newVector->name) != 0){
            q=p;
            p=p->next;
        }
        if (p && strcmp(p->name,newVector->name)==0){
            printf("Same Vector!\n");
        }
        else{
            q->next = newVector;
        }
    }
}
//PlaceMatrix
void placeMatrix(Files *files , Matrix *newMatrix){

    if (files->headMat == NULL){
        files->headMat = newMatrix;
    }
    else{
        Matrix *p = files->headMat, *q;

        while (p && strcmp(p->name,newMatrix->name) != 0){
            q=p;
            p=p->next;
        }
        if (p && strcmp(p->name,newMatrix->name)==0){
            printf("Same Matrix!\n");
        }
        else{
            q->next = newMatrix;
        }
    }
}
```

3.i) newObject function creates newFile, newMatrix and newVector.



```c
#include "functions.h"

//NewFiles
Files* newFiles(){

    Files* files = (Files*) malloc (sizeof(Files));
    files->headMat = NULL;
    files->headVec = NULL;
    return files;
}
//NewVector
Vector* newVec(){

    Vector *newVector = (Vector *) malloc (sizeof(Vector));
    newVector->next = NULL;
    return newVector;
}
//NewMatrix
Matrix* newMat(){

    Matrix *newMatrix = (Matrix *) malloc (sizeof(Matrix));
    newMatrix->next = NULL;
    return newMatrix;
}
```