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

int* allocateMatrix(int rows, int cols){
    int*matrix = (int*) malloc(rows * cols * sizeof(int));
    return matrix;
}

void freemat(int* matrix){
    free(matrix);
}

void multiplyMatrices(int* matrixA, int* matrixB, int* matrixC, int rowsA, int colsA, int colsB){
    for(int i=0;i<rowsA;i++){
        for(int j = 0; j < colsB; j++){
            matrixC[i * colsB +j] = 0;
            for(int k = 0; k < colsA; k++) {
                matrixC[i * colsB + j] += matrixA[i * colsA + k] * matrixB[k * colsB + j];
            }
        }
    }
}


void printMatrix(int* matrix, int rows, int cols) {
    for (int i = 0; i < rows; i++){
        for (int j = 0; j <cols; j++){
            printf("%d", matrix[i * cols + j]);
        }
        printf("\n");
    }
}


int main(){
    int rowsA, colsA, rowsB, colsB;

    printf("Enter number of rows for matrix A:");
    scanf("%d", &rowsA);
    printf("Enter number of columns for matrix A:");
    scanf("%d", &colsA);
    printf("Enter number of rows for matrix B:");
    scanf("%d", &rowsB);
    printf("Enter number of columns for matrix B:");
```

```
    scanf("%d", &colsB);

    if (colsA != rowsB){
        printf("Matrices cannot be multiplied.\n");
        return 1;
    }

    int* matrixA = allocateMatrix(rowsA, colsA);
    int* matrixB = allocateMatrix(rowsB, colsB);
    int* matrixC = allocateMatrix(rowsA, colsB);

    printf("Enter elements of matrix A: \n");
    for( int i = 0; i < rowsA; i++){
        for (  int j = 0; j < colsA; j++) {
            scanf("%d", &matrixA[i * colsA + j]);
        }

    }

    printf("Enter elements of matrix B: \n");
    for( int i = 0; i < rowsB; i++){
        for (  int j = 0; j < colsB; j++) {
            scanf("%d", &matrixB[i * colsB + j]);
        }

    }

    multiplyMatrices(matrixA, matrixB, matrixC, rowsA, colsA, colsB);

    printf("Matrix A:\n");
    printMatrix(matrixA, rowsA, colsA);
    printf("Matrix B:\n");
    printMatrix(matrixB, rowsB, colsB);
    printf("Matrix C (A * B):\n");
    printMatrix(matrixC, rowsA, colsB);

    freemat(matrixA);
    freemat(matrixB);
    freemat(matrixC);

    return 0;
}
```

Output:

```
EnternumberofrowsformatrixA:3
```

```
EnternumberofcolumnsformatrixA:3
EnternumberofrowsformatrixB:3
EnternumberofcolumnsformatrixB:3
 EnterelementsofmatrixA: 1
2
 3
 4
 5
 6
7
8
9
 EnterelementsofmatrixB: 9
 8
7
6
5
4
3
2
1
MatrixA: 1 2 3
              4 5 6
              7 8 9
MatrixB: 9 8 7
             6 5 4
             3 2 1
 MatrixC(A*B): 302418
                       846954
                      13811490


#include<stdio.h>
#include<string.h>

#define MAX_STUDENTS 100
#define MAX_NAME_LENGTH 50

void createList(char students [][MAX_NAME_LENGTH], int* size) {
    printf("Enter the number of students: ");
    scanf("%d", size);
    for (int i = 0; i < *size; i++) {
        printf("Enter student name %d: ", i + 1);
        scanf("%s", students[i]);
```

```
    }
}

void insertStudent(char students[][MAX_NAME_LENGTH], int* size, int position, char* name){
    for (int i = *size; i > position; i--){
        strcpy(students[i], students[i - 1]);
    }
    strcpy(students[position], name);
    (*size)++;
}

void deleteStudent(char students[][MAX_NAME_LENGTH], int* size, int position){
    for (int  i = 0; i < *size - 1; i++)
    {
        strcpy(students[i], students[i + 1]);
    }
    (*size)--;
}

void displayList(char students[][MAX_NAME_LENGTH], int size){
    printf("Student list: [" );
    for ( int i = 0; i < size; i++){
        printf("%s", students[i]);
        if(i < size -1){
            printf(", ");
        }
    }
    printf("]\n");
}

void searchStudent(char students[][MAX_NAME_LENGTH], int size, char* name){
    int found = 0;
    for(int i = 0; i < size; i++){
        if(strcmp(students[i], name) == 0){
            printf("%s found at position %d\n", name, i);
            found = 1;
            break;
        }
    }
    if(!found){
        printf("%s not found\n", name);
    }
}
```

```
int main(){
    char students[MAX_STUDENTS][MAX_NAME_LENGTH];
    int size =0;

    while (1)
    {
        printf("1. Create the list of students\n");
        printf("2. Insert a new student\n");
        printf("3. Delete a student\n");
        printf("4. Display the student list\n");
        printf("5. Search for a student\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        int choice;
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            createList(students, &size);
            displayList(students, size);
            break;
        case 2:
            char name[MAX_NAME_LENGTH];
            printf("Enter the students's name to insert: ");
            scanf("%s", name);
            int position;
            printf("Enter the position to insert the student: ");
            scanf("%d", &position);
            insertStudent(students, &size, position, name);
            displayList(students, size);
            break;
        case 3:
            char deleteMethod;
            printf("Delete  by name or position (n/p): ");
            scanf("%c",&deleteMethod);
            if(deleteMethod == 'n'){
                char deleteName[MAX_NAME_LENGTH];
                printf("Enter the students name to delete");
                scanf("%s", deleteName);
                for(int i = 0; i < size; i++){
                    if( strcmp(students[i], deleteName) == 0){
                        deleteStudent(students, &size, i);
                        displayList(students, size);
                        break;
```

```
                }
            }
        } else{
            int deletePosition;
            printf("Enter the position to delete the student: ");
            scanf("%d", &deletePosition);
            deleteStudent(students, &size, deletePosition);
            displayList(students, size);
        }
        break;
    case 4:
        displayList(students, size);
        break;
    case 5:
        char searchName[MAX_NAME_LENGTH];
        printf("Enter the students name to search: ");
        scanf("%s", searchName);
        searchStudent(students, size, searchName);
        break;
    case 6:
        printf("Exiting the program...\n");
        return 0;
    }

  }
 return 0;
}
```