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

#define ROWS 2
#define COLS 2

struct MenuItem
{
    char itemName[50];
    double itemPrice;
};

struct CanteenManagement
{
    struct MenuItem menuA[ROWS][COLS];
    struct MenuItem menuB[ROWS][COLS];
    struct MenuItem resultMatrixAdd[ROWS][COLS];
    struct MenuItem resultMatrixMultiply[ROWS][COLS];
};

void displayMatrix(struct MenuItem matrix[][COLS])
{
    printf("Matrix:\n");
    for (int i = 0; i < ROWS; i++)
    {
        for (int j = 0; j < COLS; j++)
        {
            printf("%s(%.2f)\t", matrix[i][j].itemName, matrix[i][j].itemPrice);
        }
        printf("\n");
    }
}

void addMatrices(struct MenuItem matrixA[][COLS], struct MenuItem
matrixB[][COLS], struct MenuItem resultMatrix[][COLS])
{
    for (int i = 0; i < ROWS; i++)
    {
        for (int j = 0; j < COLS; j++)
        {
            strcpy(resultMatrix[i][j].itemName, "Result");
            resultMatrix[i][j].itemPrice = matrixA[i][j].itemPrice +
matrixB[i][j].itemPrice;
```

```
        }
    }
}

void multiplyMatrices(struct MenuItem matrixA[][COLS], struct MenuItem
matrixB[][COLS], struct MenuItem resultMatrix[][COLS])
{
    for (int i = 0; i < ROWS; i++)
    {
        for (int j = 0; j < COLS; j++)
        {
            strcpy(resultMatrix[i][j].itemName, "Result");
            resultMatrix[i][j].itemPrice = matrixA[i][j].itemPrice *
matrixB[i][j].itemPrice;
        }
    }
}

void addItem(struct MenuItem matrix[][COLS], int row, int col, const char
itemName[], double itemPrice)
{
    strcpy(matrix[row][col].itemName, itemName);
    matrix[row][col].itemPrice = itemPrice;
}

void removeItem(struct MenuItem matrix[][COLS], int row, int col)
{
    strcpy(matrix[row][col].itemName, "");
    matrix[row][col].itemPrice = 0.0;
}

void searchItem(struct MenuItem matrix[][COLS], int row, int col, const char
itemName[])
{
    for (int i = 0; i < ROWS; i++)
    {
        for (int j = 0; j < COLS; j++)
        {
            if (strcmp(matrix[i][j].itemName, itemName) == 0)
            {
                printf("Item found: %s, Price: %.2f\n", matrix[i][j].itemName,
matrix[i][j].itemPrice);
                return;
            }
        }
```

```
    }
    printf("Item not found in the menu.\n");
}

void fillMatrix(struct MenuItem matrix[][COLS])
{
    printf("Enter values for the matrix:\n");
    for (int i = 0; i < ROWS; i++)
    {
        for (int j = 0; j < COLS; j++)
        {
            printf("Enter name for element [%d][%d]: ", i + 1, j + 1);
            scanf("%s", matrix[i][j].itemName);
            printf("Enter price for element [%d][%d]: ", i + 1, j + 1);
            scanf("%lf", &matrix[i][j].itemPrice);
        }
    }
}

int main()
{
    struct CanteenManagement canteen;

    printf("Enter values for Matrix A:\n");
    fillMatrix(canteen.menuA);

    printf("\nEnter values for Matrix B:\n");
    fillMatrix(canteen.menuB);

    printf("\nMatrix A:\n");
    displayMatrix(canteen.menuA);

    printf("\nMatrix B:\n");
    displayMatrix(canteen.menuB);

    int choice, row, col;
    char itemName[50];
    double itemPrice;
    char exitInput[5];

    while (1) // Infinite loop
    {
        printf("\nMenu Item Operations:\n");
        printf("1. Add Item\n");
        printf("2. Remove Item\n");
```

```
        printf("3. Search Item\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        if (choice == 4)
        {
            printf("Do you really want to exit? (yes/no): ");
            scanf("%s", exitInput);
            if (strcmp(exitInput, "yes") == 0)
            {
                printf("Exiting the program.\n");
                break;
            }
            else
            {
                continue;
            }
        }

        switch (choice)
        {
        case 1:
            printf("Enter row and column for the new item: ");
            scanf("%d %d", &row, &col);
            printf("Enter name for the new item: ");
            scanf("%s", itemName);
            printf("Enter price for the new item: ");
            scanf("%lf", &itemPrice);
            addItem(canteen.menuA, row - 1, col - 1, itemName, itemPrice);
            break;

        case 2:
            printf("Enter row and column to remove the item: ");
            scanf("%d %d", &row, &col);
            removeItem(canteen.menuA, row - 1, col - 1);
            break;

        case 3:
            printf("Enter name of the item to search: ");
            scanf("%s", itemName);
            searchItem(canteen.menuA, ROWS, COLS, itemName);
            break;

        default:
```

```
            printf("Invalid choice\n");
        }

        printf("\nUpdated Matrix A:\n");
        displayMatrix(canteen.menuA);
    }

    return 0;
}
```

Output- matrix operation

```
Enter values for Matrix A:
Enter values for the matrix:
Enter name for element [1][1]: pizza
Enter price for element [1][1]: 120
Enter name for element [1][2]: burger
Enter price for element [1][2]: 80
Enter name for element [2][1]: momo
Enter price for element [2][1]: 80
Enter name for element [2][2]: fries
Enter price for element [2][2]: 120

Enter values for Matrix B:
Enter values for the matrix:
Enter name for element [1][1]: biryani
Enter price for element [1][1]: 160
Enter name for element [1][2]: paneertikka
Enter price for element [1][2]: 200
Enter name for element [2][1]: chickentikka
Enter price for element [2][1]: 50
Enter name for element [2][2]: icecream
Enter price for element [2][2]: 80
```

```
Matrix A:
Matrix:
pizza(120.00)   burger(80.00)
momo(80.00)     fries(120.00)

Matrix B:
Matrix:
biryani(160.00) paneertikka(200.00)
chickentikka(50.00)     icecream(80.00)

Matrix Addition Result:
Matrix:
Result(280.00)  Result(280.00)
Result(130.00)  Result(200.00)

Matrix Multiplication Result:
Matrix:
Result(19200.00)        Result(16000.00)
Result(4000.00) Result(9600.00)
```

Insertion

```
Menu Item Operations:
1. Add Item
2. Remove Item
3. Search Item
Enter your choice:
1
Enter row and column for the new item: 1 1
Enter name for the new item: friedrice
Enter price for the new item: 120

Updated Matrix A:
Matrix:
friedrice(120.00)       burger(80.00)
momo(80.00)     fries(120.00)
PS C:\Users\nayan raj\OneDrive\Desktop\mca_1_B\dsa\lab ex-1>
```

Deletion

```
Menu Item Operations:
1. Add Item
2. Remove Item
3. Search Item
4. Exit
Enter your choice: 2
Enter row and column to remove the item: 1 2

Updated Matrix A:
Matrix:
friedrice(120.00)       (0.00)
momo(80.00)     fries(120.00)
```

Searching

```
Menu Item Operations:
1. Add Item
2. Remove Item
3. Search Item
4. Exit
Enter your choice: 3
Enter name of the item to search: pizza
Item not found in the menu.

Updated Matrix A:
Matrix:
friedrice(120.00)       (0.00)
momo(80.00)     fries(120.00)
```