```c
//  main.c
//  CS143 - 7th project
//  Created by Mark Antonio on 4/5/20.
//  Copyright © 2020 Mark Antonio. All rights reserved.
//
#include <stdio.h>
//main menu
void MainMenuOptions(){
printf("**----------------------------------**\n");
printf("**Please pick of the following options**\n");
printf("**----------------------------------**\n");
printf("1. Preview the Branch\\Sales matrix\n");
printf("2. Add a new branch\n");
printf("3. Delete an existing branch\n");
printf("4. Calculate total sales\n");
printf("5. Calculate percentage share of each branch\n");
printf("6. Determine the month of the peak sales\n");
printf("7. Display sales of a specific month\n");
printf("8. Display sales of a specific branch\n");
printf("0. Quit\n");
printf("You choice: ");
}
//*********************************************************************
 *******************************************//
//Functionality 1 [Printing]
void MatrixPrint(float arr[][12], int countOfBranches){
    printf("Branch\\Month:");//header of the table
    for (int i=0; i<12; i++) {//header of the table (Cont.)
        printf("%9d",i+1);
    }
    printf("\n");
    for(int x=0;x<countOfBranches;x++){
        printf("Branch %d:        ",x+1);
        for (int y=0; y<12; y++) {
            printf("%6.1f   ",arr[x][y]);
        }
        printf("\n");
    }
}
//*********************************************************************
 *******************************************//
//Functionality 2 [Add]
void MatrixAddition(float arr[][12], int countOfBranches){
    for(int i=0;i<12;i++){
        printf("sales for month %d:",i+1);
        scanf("%f",&arr[countOfBranches][i]);
    }
}
//*********************************************************************
 *******************************************//
//Functionality 3 [Delete]
void MatrixDeletion(float arr[][12], int *countOfBranches){
    int oldCountOfBranches = *countOfBranches;
    int newCountOfBranches;
    int targetToBeDeleted;
    printf("Input the number of Branch you want to delete: ");
    scanf("%d",&targetToBeDeleted);
    if (oldCountOfBranches==1) { //only one branch data exists
        printf("[Warning!]You have cleared the app data\n");
        newCountOfBranches=0;
        *countOfBranches=newCountOfBranches;
    }else if (oldCountOfBranches==2){ //we can only shrink the array in
     this case, he wants to delete the second row
        if (targetToBeDeleted==2) {
            newCountOfBranches=1;
            *countOfBranches=newCountOfBranches;
        }else{//he wants to delete the first row
            for (int i=0; i<12; i++) {//we shift the second cell up
                arr[0][i] = arr[1][i];
            }
            newCountOfBranches=1;
            *countOfBranches=newCountOfBranches;
        }
    }else if (oldCountOfBranches>2){ //we have more than 2 branches (swap
     mechanics)
        if (targetToBeDeleted==1) { //he wants to delete the first row
            for (int i=0; i<12; i++) {//we shift the second cell up
                arr[0][i] = arr[oldCountOfBranches-1][i];
            }
            newCountOfBranches=oldCountOfBranches-1;
            *countOfBranches=newCountOfBranches;
        }else if (targetToBeDeleted==oldCountOfBranches){ //he wants to
         delete the last row, so we use the shrink mechanism discarding the
         last line

            newCountOfBranches=oldCountOfBranches-1;
            *countOfBranches=newCountOfBranches;
        }else{ //he neither wants to delete the first or the last, so we
         use swap and shrink
            for (int i=0; i<12; i++) {//we shift the second cell up
                arr[targetToBeDeleted-1][i] = arr[oldCountOfBranches-1][i];
                 //swap
            }
            newCountOfBranches=oldCountOfBranches-1; //shrink
            *countOfBranches=newCountOfBranches;
        }
    }
}
//****************************************************************************
 ******************************************//
//Functionality 4 [Total sales]
float TotalSales(float arr[][12],int countOfBranches){
    float totalSales=0;
    for (int i=0; i<countOfBranches; i++) {
        for (int j=0; j<12; j++) {
            totalSales+=arr[i][j];
        }
    }
    return totalSales;
}
```

```c
//*****************************************************************************
 ******************************************//
//Functionality 5 [sales share]
void SalesShare(float arr[][12],int countOfBranches,int totalSales){
    for (int i=0; i<countOfBranches; i++) {
        float sum=0;
        for (int j=0; j<12; j++) {
            sum+=arr[i][j];
        }
        printf("Branch %d:        ",i+1);
        printf("%6.1f%%\n",(sum*100)/totalSales);
    }
}
//*****************************************************************************
 ******************************************//
//Functionality 6 [Peak sales]
void PeakSales(float arr[][12],int countOfBranches){
    float peak=0;
    int monthOfPeak=0;
    for (int i=0; i<12; i++) {
        int sum=0;
        for (int j=0; j<countOfBranches; j++) {
            sum+=arr[j][i];
            if (peak<sum) {
                peak = sum;
                monthOfPeak = i+1;
            }
        }
    }
    printf("[Information]Month %d has the peak sales of %0.1f
     EGP\n",monthOfPeak,peak);
}
//*****************************************************************************
 ******************************************//
//Functionality 7 [sales of a specific month]
void MonthSales(float arr[][12],int countOfBranches){
    int targetMonth;
    printf("Which month? ");
    scanf("%d",&targetMonth);
    float tempArray[countOfBranches]; //array of branches sales
    float tempArrayPositions[countOfBranches]; //array for positions
    for (int i=1; i<=countOfBranches; i++) {
        tempArrayPositions[i-1]=i;
    }
    for (int i=0; i<countOfBranches; i++) {
        tempArray[i]=arr[i][targetMonth-1];
    }
    for(int i=1;i<countOfBranches;i++){ //passes
        for(int j=0;j<countOfBranches-i;j++){ //positions
            if(tempArray[j]<tempArray[j+1]){
                int temp = tempArray[j]; //sorting the sales values
                tempArray[j] = tempArray[j+1];
                tempArray[j+1] = temp;
                int temp2 = tempArrayPositions[j]; //the positions array is
                 going alongside it
                tempArrayPositions[j] = tempArrayPositions[j+1];
                tempArrayPositions[j+1] = temp2;
            }
        }
    }
    for (int i=0; i<countOfBranches; i++) {
        printf("Branch %d:  %0.1f \n",(int)tempArrayPositions[i],
         tempArray[i]);
    }
}
//****************************************************************************
 ******************************************//
//Functionality 8 [sales of a specific branch]
void BranchSales(float arr[][12],int countOfBranches){
    int targetBranch;
    printf("Which branch? ");
    scanf("%d",&targetBranch);
    float salesValues[12];
    float salesValuesPositions[12];

    for (int i=0; i<12; i++) {
        salesValuesPositions[i]=i+1;
    }

    for (int i=0; i<12; i++) {
        salesValues[i]=arr[targetBranch-1][i];
    }

    for (int x=0; x<11; x++) {
        int indexOfmax=x;
        for (int y=x+1; y<12; y++) {
            if (salesValues[y]>salesValues[indexOfmax]) {
                indexOfmax=y;
            }
        }
        if (indexOfmax!=x) {
            int temp = salesValues[x]; //sorting the sales values
            salesValues[x] = salesValues[indexOfmax];
            salesValues[indexOfmax] = temp;

            int temp2 = salesValuesPositions[x]; //sorting the sales values
            salesValuesPositions[x] = salesValuesPositions[indexOfmax];
            salesValuesPositions[indexOfmax] = temp2;
        }
    }
   for (int i=0; i<12; i++) {
    printf("Month %d:  %0.1f \n",(int)salesValuesPositions[i],
     salesValues[i]);
    }
}
//****************************************************************************
 ******************************************//
int main(int argc, const char * argv[]) {
```

```c
int countOfBranches;
printf("*****Welcome to the retail analysis app*****\n");
printf("Kindly specify the count of branches: ");
scanf("%d",&countOfBranches);
//creating the data matrix (2-D Array)
float mainDataMatrix[countOfBranches+99][12]; //12 for a complete year
//**********************************************************************
 *****************************//
//filling the matrix
for (int i=0; i<countOfBranches; i++) {
    printf("Sales for branch %d:\n",i+1);
    for (int j=0; j<12; j++) {
        printf("sales for month %d:",j+1);
        scanf("%f",&mainDataMatrix[i][j]);
    }
    printf("\n");
}
//**********************************************************************
 *****************************//
//prompt menu
int userChoice;
float totalSales;
do {
    MainMenuOptions();
    scanf("%d",&userChoice);
    switch (userChoice) {
        case 1:
            MatrixPrint(mainDataMatrix, countOfBranches);
            break;
        case 2:
            MatrixAddition(mainDataMatrix, countOfBranches);
            MatrixPrint(mainDataMatrix, countOfBranches+1);
            countOfBranches+=1; //this is an addition function
            break;
        case 3:
            printf("\n[Information!]Current count of branches is:
             %d\n", countOfBranches);
            MatrixDeletion(mainDataMatrix, &countOfBranches);
            MatrixPrint(mainDataMatrix, countOfBranches);
            break;
        case 4:
            totalSales = TotalSales(mainDataMatrix, countOfBranches);
            printf("[Information]Total sales: %0.1f EGP\n",totalSales);
            break;
        case 5:
            totalSales = TotalSales(mainDataMatrix, countOfBranches);
             //this assignment is a debug to avoid errors if totalSales
             wasnt initialized
            SalesShare(mainDataMatrix, countOfBranches, totalSales);
            break;
        case 6:
            PeakSales(mainDataMatrix, countOfBranches);
            break;
        case 7:
            MonthSales(mainDataMatrix, countOfBranches);
```

```
                break;
            case 8:
                BranchSales(mainDataMatrix, countOfBranches);
            default:
                break;
        }
    } while (userChoice!=0);
    return 0;
}
```