Roll NO:27

NAME : Mali Anjali Prakash

DIV: FYMCA-A

Assignment no -11

#include <stdio.h>

#include <string.h>

#define MAX\_SALES 100

// Structure to represent sales data

typedef struct {

    int salesID;

    char customerName[50];

    char productName[50];

    int quantitySold;

    float totalSaleAmount;

} Sale;

// Function to merge two halves in Merge Sort

void merge(Sale sales[], int left, int mid, int right) {

    int n1 = mid - left + 1;

    int n2 = right - mid;

    // Temporary arrays

    Sale leftArray[n1], rightArray[n2];

    // Copy data to temporary arrays

    for (int i = 0; i < n1; i++)

        leftArray[i] = sales[left + i];

    for (int i = 0; i < n2; i++)

        rightArray[i] = sales[mid + 1 + i];

    int i = 0, j = 0, k = left;

    // Merge the two arrays

    while (i < n1 && j < n2) {

        if (leftArray[i].totalSaleAmount > rightArray[j].totalSaleAmount) {

            sales[k] = leftArray[i];

            i++;

        } else {

            sales[k] = rightArray[j];

            j++;

        }

        k++;

    }

    // Copy remaining elements from leftArray, if any

    while (i < n1) {

        sales[k] = leftArray[i];

        i++;

        k++;

    }

    // Copy remaining elements from rightArray, if any

    while (j < n2) {

        sales[k] = rightArray[j];

        j++;

        k++;

    }

}

// Merge Sort function to sort sales by total sale amount in descending order

void mergeSort(Sale sales[], int left, int right) {

    if (left < right) {

        int mid = left + (right - left) / 2;

        // Recursively sort the two halves

        mergeSort(sales, left, mid);

        mergeSort(sales, mid + 1, right);

        // Merge the sorted halves

        merge(sales, left, mid, right);

    }

}

// Function to perform partitioning for Quick Sort

int partition(Sale sales[], int low, int high) {

    float pivot = sales[high].totalSaleAmount;

    int i = low - 1;

    for (int j = low; j < high; j++) {

        if (sales[j].totalSaleAmount > pivot) {

            i++;

            Sale temp = sales[i];

            sales[i] = sales[j];

            sales[j] = temp;

        }

    }

    Sale temp = sales[i + 1];

    sales[i + 1] = sales[high];

    sales[high] = temp;

    return (i + 1);

}

// Quick Sort function to sort sales by total sale amount in descending order

void quickSort(Sale sales[], int low, int high) {

    if (low < high) {

        int pi = partition(sales, low, high);

        // Recursively sort the two parts

        quickSort(sales, low, pi - 1);

        quickSort(sales, pi + 1, high);

    }

}

// Function to display the sales data

void displaySales(Sale sales[], int n) {

    printf("\nSales ID\tCustomer Name\tProduct Name\tQuantity Sold\tTotal Sale Amount\n");

    printf("-------------------------------------------------------------------------------\n");

    for (int i = 0; i < n; i++) {

        printf("%d\t\t%s\t\t%s\t\t%d\t\t%.2f\n",

            sales[i].salesID,

            sales[i].customerName,

            sales[i].productName,

            sales[i].quantitySold,

            sales[i].totalSaleAmount);

    }

}

int main() {

    int n;

    // Input the number of sales records

    printf("Enter number of sales records: ");

    scanf("%d", &n);

    Sale sales[MAX\_SALES];

    // Input sales data

    for (int i = 0; i < n; i++) {

        printf("\nEnter details for sale %d:\n", i + 1);

        printf("Sales ID: ");

        scanf("%d", &sales[i].salesID);

        getchar();  // Consume the newline character left by scanf

        printf("Customer Name: ");

        fgets(sales[i].customerName, 50, stdin);

        sales[i].customerName[strcspn(sales[i].customerName, "\n")] = '\0';  // Remove newline character

        printf("Product Name: ");

        fgets(sales[i].productName, 50, stdin);

        sales[i].productName[strcspn(sales[i].productName, "\n")] = '\0';  // Remove newline character

        printf("Quantity Sold: ");

        scanf("%d", &sales[i].quantitySold);

        printf("Total Sale Amount: ");

        scanf("%f", &sales[i].totalSaleAmount);

    }

    // Sort using Merge Sort

    printf("\nSorting sales data using Merge Sort...\n");

    mergeSort(sales, 0, n - 1);

    displaySales(sales, n);

    // Sort using Quick Sort

    printf("\nSorting sales data using Quick Sort...\n");

    quickSort(sales, 0, n - 1);

    displaySales(sales, n);

    return 0;

}



