STRUCTURES and POINTERS

Q.NO:1

CODE:

#include <stdio.h>

struct Type {

char type[20];

char commonFriends[50];

char placesVisited[100];

};

struct Friends {

char name[50];

char petName[20];

char phoneNumber[15];

struct Type friendType;

};

int main() {

struct Friends friends[3];

int i;

for (i = 0; i < 3; ++i) {

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

printf("Name: ");

scanf(" %[^\n]s", friends[i].name);

printf("Pet Name: ");

scanf(" %[^\n]s", friends[i].petName);

printf("Phone Number: ");

scanf(" %[^\n]s", friends[i].phoneNumber);

printf("Type of Friend (School/College/Area): ");

scanf(" %[^\n]s", friends[i].friendType.type);

printf("Names of Common Friends: ");

scanf(" %[^\n]s", friends[i].friendType.commonFriends);

printf("Places Visited Together: ");

scanf(" %[^\n]s", friends[i].friendType.placesVisited);

printf("\n");

}

// Displaying details in proper format

printf("Friend Details:\n");

for (i = 0; i < 3; ++i) {

printf("Friend %d\n", i + 1);

printf("Name: %s\n", friends[i].name);

printf("Pet Name: %s\n", friends[i].petName);

printf("Phone Number: %s\n", friends[i].phoneNumber);

printf("Type of Friend: %s\n", friends[i].friendType.type);

printf("Common Friends: %s\n", friends[i].friendType.commonFriends);

printf("Places Visited Together: %s\n", friends[i].friendType.placesVisited);

printf("\n");

}

return 0;

}

Q.no:2

CODE:

#include <stdio.h>

#include <string.h>

#define NUM\_PRODUCTS 5

typedef struct {

char name[50];

int id;

float price;

} Product;

void inputProducts(Product \*products, int numProducts);

void displayProducts(Product \*products, int numProducts);

float calculateTotalCost(Product \*products, int numProducts);

Product findMostExpensiveProduct(Product \*products, int numProducts);

Product findLeastExpensiveProduct(Product \*products, int numProducts);

int main() {

Product products[NUM\_PRODUCTS];

float totalCost;

Product mostExpensive, leastExpensive;

inputProducts(products, NUM\_PRODUCTS);

printf("\nProduct Details:\n");

displayProducts(products, NUM\_PRODUCTS);

totalCost = calculateTotalCost(products, NUM\_PRODUCTS);

printf("\nTotal cost of all products: $%.2f\n", totalCost);

mostExpensive = findMostExpensiveProduct(products, NUM\_PRODUCTS);

leastExpensive = findLeastExpensiveProduct(products, NUM\_PRODUCTS);

printf("\nMost Expensive Product:\n");

printf("Name: %s\nID: %d\nPrice: $%.2f\n", mostExpensive.name, mostExpensive.id, mostExpensive.price);

printf("\nLeast Expensive Product:\n");

printf("Name: %s\nID: %d\nPrice: $%.2f\n", leastExpensive.name, leastExpensive.id, leastExpensive.price);

return 0;

}

void inputProducts(Product \*products, int numProducts) {

for (int i = 0; i < numProducts; i++) {

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

printf("Name: ");

scanf(" %[^\n]s", products[i].name); // Read a line of text

printf("ID: ");

scanf("%d", &products[i].id);

printf("Price: ");

scanf("%f", &products[i].price);

printf("\n");

}

}

void displayProducts(Product \*products, int numProducts) {

for (int i = 0; i < numProducts; i++) {

printf("Product %d:\n", i + 1);

printf("Name: %s\n", products[i].name);

printf("ID: %d\n", products[i].id);

printf("Price: $%.2f\n\n", products[i].price);

}

}

float calculateTotalCost(Product \*products, int numProducts) {

float totalCost = 0.0;

for (int i = 0; i < numProducts; i++) {

totalCost += products[i].price;

}

return totalCost;

}

Product findMostExpensiveProduct(Product \*products, int numProducts) {

Product mostExpensive = products[0];

for (int i = 1; i < numProducts; i++) {

if (products[i].price > mostExpensive.price) {

mostExpensive = products[i];

}

}

return mostExpensive;

}

Product findLeastExpensiveProduct(Product \*products, int numProducts) {

Product leastExpensive = products[0];

for (int i = 1; i < numProducts; i++) {

if (products[i].price < leastExpensive.price) {

leastExpensive = products[i];

}

}

return leastExpensive;

}