# Student Information:

* + Define a structure to store student information, including name, roll number, and marks in three subjects.
  + Write a program to input data for 5 students and display the details along with their average marks.

#include <stdio.h>

information struct Student {

char name[50]; int rollNumber;

float marks[3]; // Marks for three subjects

};

int main() {

struct Student students[5]; // Array to store data for 5 students float averageMarks;

// Input data for 5 students for (int i = 0; i < 5; i++) {

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

printf("Name: ");

scanf(" %[^\n]s", students[i].name); // Space before % prevents newline issues

printf("Roll Number: ");

scanf("%d", &students[i].rollNumber);

printf("Enter marks in 3 subjects: "); for (int j = 0; j < 3; j++) {

scanf("%f", &students[i].marks[j]);

}

printf("\n");

}

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

for (int i = 0; i < 5; i++) { float total = 0;

printf("\nName: %s\n", students[i].name); printf("Roll Number: %d\n", students[i].rollNumber);

printf("Marks: %.2f, %.2f, %.2f\n", students[i].marks[0], students[i].marks[1], students[i].marks[2]);

marks for (int j = 0; j < 3; j++) {

total += students[i].marks[j];

}

averageMarks = total / 3;

printf("Average Marks: %.2f\n", averageMarks);

}

return 0;

}

O/p: Enter details for student 1: Name: likitha

Roll Number: 69

Enter marks in 3 subjects: 80 90 75

Enter details for student 2: Name: pooja

Roll Number: 70

Enter marks in 3 subjects: 67 54 68

Enter details for student 3: Name: sony

Roll Number: 71

Enter marks in 3 subjects: 78 67 85

Enter details for student 4: Name: kane

Roll Number: 72

Enter marks in 3 subjects: 89 90 67

Enter details for student 5: Name: ram

Roll Number: 73

Enter marks in 3 subjects: 95 77 77 88

Student Details:

Name: likitha Roll Number: 69

Marks: 80.00, 90.00, 75.00

Average Marks: 81.67

Name: pooja Roll Number: 70

Marks: 67.00, 54.00, 68.00

Average Marks: 63.00

Name: sony Roll Number: 71

Marks: 78.00, 67.00, 85.00

Average Marks: 76.67

Name: kane Roll Number: 72

Marks: 89.00, 90.00, 67.00

Average Marks: 82.00

Name: ram

Roll Number: 73

Marks: 95.00, 77.00, 88.00

Average Marks: 86.67

# Employee Details:

* + Create a structure to store employee details like name, ID, salary, and department.
  + Write a function to display the details of employees whose salary is above a certain threshold.

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

// Employee Details struct Employee {

char name[50]; int id;

float salary;

char department[30];

};

void displayHighSalaryEmployees(struct Employee employees[], int size, float threshold) {

printf("Employees with salary above %.2f:\n", threshold); for (int i = 0; i < size; i++) {

if (employees[i].salary > threshold) {

printf("Name: %s, ID: %d, Salary: %.2f, Department: %s\n", employees[i].name, employees[i].id, employees[i].salary,

employees[i].department);

}

}

}

int main() {

struct Employee employees[3] = {

{"Likitha", 1, 50000, "HR"},

{"kane", 2, 60000, "Engineering"},

{"ram", 3, 40000, "Marketing"}

};

float threshold = 45000; displayHighSalaryEmployees(employees, 3, threshold);

return 0;

}

O/p:

Employees with salary above 45000.00:

Name: likitha, ID: 1, Salary: 50000.00, Department: HR Name: kane, ID: 2, Salary: 60000.00, Department: Engineering

# Book Store Inventory:

* + Define a structure to represent a book with fields for title, author, ISBN, and price.
  + Write a program to manage an inventory of books and allow searching by title.

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

struct Book { char title[100];

char author[50]; float price;

};

int main() {

struct Book books[3] = {

{"C Programming", "Dennis", 25.50},

{"Data Structures", "Tanenbaum", 30.00},

{"Algorithms", "Sedgewick", 35.75}

};

char searchTitle[100];

printf("Enter book title to search: "); scanf("%s", searchTitle);

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

if (strchr(books[i].title, searchTitle) == 0) {

printf("Book Found: %s by %s, Price: %.2f\n", books[i].title, books[i].author, books[i].price);

return 0;

}

}

printf("Book not found.\n"); return 0;

}

O/p:

Enter book title to search: Alogorithms

Book Found: Algorithms by Sedgewick, Price: 35.75

# Date Validation:

* + Create a structure to represent a date with day, month, and year.
  + Write a function to validate if a given date is correct (consider leap years).

Sol: #include <stdio.h>

struct Date { int day; int month; int year;

};

int isValidDate(struct Date date) {

if (date.month < 1 || date.month > 12) return 0;

int daysInMonth[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

if (date.year % 4 == 0 && (date.year % 100 != 0 || date.year % 400 == 0)) { daysInMonth[1] = 29; // Leap year

}

return date.day >= 1 && date.day <= daysInMonth[date.month - 1];

}

int main() {

struct Date date = {29, 2, 2024}; // Example date if (isValidDate(date)) {

printf("Date is valid.\n");

} else {

printf("Date is invalid.\n");

}

return 0;

}

O/p:

Date is valid.

# Complex Numbers:

* + Define a structure to represent a complex number with real and imaginary parts.
  + Implement functions to add, subtract, and multiply two complex numbers.

Sol: #include <stdio.h>

// Define a structure to represent a complex number struct Complex {

float real;

float imaginary;

};

int main() {

struct Complex c1, c2, result;

// Input for the first complex number

printf("Enter first complex number (real and imaginary parts): ");

scanf("%f %f", &c1.real, &c1.imaginary);

// Input for the second complex number

printf("Enter second complex number (real and imaginary parts): "); scanf("%f %f", &c2.real, &c2.imaginary);

// Addition of two complex numbers result.real = c1.real + c2.real;

result.imaginary = c1.imaginary + c2.imaginary; printf("Addition: %.2f + %.2fi\n", result.real, result.imaginary);

// Subtraction of two complex numbers result.real = c1.real - c2.real;

result.imaginary = c1.imaginary - c2.imaginary; printf("Subtraction: %.2f + %.2fi\n", result.real, result.imaginary);

// Multiplication of two complex numbers

result.real = (c1.real \* c2.real) - (c1.imaginary \* c2.imaginary); result.imaginary = (c1.real \* c2.imaginary) + (c1.imaginary \* c2.real); printf("Multiplication: %.2f + %.2fi\n", result.real, result.imaginary);

return 0;

}

O/p: Enter first complex number (real and imaginary parts): 2 5 Enter second complex number (real and imaginary parts): 6 8 Addition: 8.00 + 13.00i

Subtraction: -4.00 + -3.00i Multiplication: -28.00 + 46.00i

# Bank Account:

* + Design a structure to store information about a bank account, including account number, account holder name, and balance.
  + Write a function to deposit and withdraw money, and display the updated balance.

Sol: #include <stdio.h>

struct BankAccount { int accountNumber; char holderName[50]; float balance;

};

int main() {

struct BankAccount account = {12345, "John Doe", 5000.0}; float depositAmount, withdrawAmount;

printf("Initial Balance: %.2f\n", account.balance);

printf("Enter deposit amount: "); scanf("%f", &depositAmount); account.balance += depositAmount;

printf("Enter withdrawal amount: "); scanf("%f", &withdrawAmount);

if (account.balance >= withdrawAmount) { account.balance -= withdrawAmount;

} else {

printf("Insufficient balance.\n");

}

printf("Updated Balance: %.2f\n", account.balance); return 0;

}

O/p: Initial Balance: 5000.00 Enter deposit amount: 25000 Enter withdrawal amount: 2000

Updated Balance: 28000.00

# Car Inventory System:

* + Create a structure for a car with fields like make, model, year, and price.
  + Write a program to store details of multiple cars and print cars within a specified price range.

Sol: #include <stdio.h>

struct Car {

char make[50]; int year;

float price;

};

int main() {

struct Car cars[3] = {

{"Toyota", 2020, 20000},

{"Honda", 2021, 25000},

{"Ford", 2019, 15000}

};

float minPrice, maxPrice; printf("Enter min and max price: ");

scanf("%f %f", &minPrice, &maxPrice);

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

if (cars[i].price >= minPrice && cars[i].price <= maxPrice) { printf("Car: %s, Year: %d, Price: %.2f\n", cars[i].make, cars[i].year,

cars[i].price);

}

}

return 0;

}

O/p:

Enter min and max price: 20000 25000 Car: Toyota, Year: 2020, Price: 20000.00

Car: Honda, Year: 2021, Price: 25000.00

# Library Management:

* + Define a structure for a library book with fields for title, author, publication year, and status (issued or available).
  + Write a function to issue and return books based on their status. Sol: #include <stdio.h>

#include <string.h>

struct Book {

char title[100];

char status[10]; // "available" or "issued"

};

int main() {

struct Book book = {"The C Programming", "available"};

printf("Current Status: %s\n", book.status);

if (strcmp(book.status, "available") == 0) { strcpy(book.status, "issued"); printf("Book '%s' issued.\n", book.title);

} else {

printf("Book is already issued.\n");

}

printf("Current Status: %s\n", book.status);

if (strcmp(book.status, "issued") == 0) { strcpy(book.status, "available"); printf("Book '%s' returned.\n", book.title);

}

return 0;

}

O/p:

Current Status: available

Book 'The C Programming' issued. Current Status: issued

Book 'The C Programming' returned.

# Student Grades:

* + Create a structure to store a student's name, roll number, and an array of grades.
  + Write a program to calculate and display the highest, lowest, and average grade for each student.

Sol: #include <stdio.h>

struct Student { char name[50]; float grades[5];

};

int main() {

struct Student student = {"John", {90, 85, 92, 88, 76}};

float highest = student.grades[0], lowest = student.grades[0], sum = 0;

for (int i = 0; i < 5; i++) {

if (student.grades[i] > highest) highest = student.grades[i]; if (student.grades[i] < lowest) lowest = student.grades[i]; sum += student.grades[i];

}

printf("Highest Grade: %.2f, Lowest Grade: %.2f, Average: %.2f\n", highest, lowest, sum / 5);

return 0;

}

O/p:

Highest Grade: 92.00, Lowest Grade: 76.00, Average: 86.20

# Product Catalog:

* + Define a structure to represent a product with fields for product ID, name, quantity, and price.
  + Write a program to update the quantity of products after a sale and calculate the total sales value.

Sol: #include <stdio.h>

struct Product { int productId; char name[50];

int quantity; float price;

};

int main() {

struct Product product = {101, "Laptop", 50, 800.0}; int soldQuantity;

printf("Current Quantity: %d\n", product.quantity); printf("Enter quantity to sell: ");

scanf("%d", &soldQuantity);

if (product.quantity >= soldQuantity) { product.quantity -= soldQuantity;

printf("Sold %d units. Updated quantity: %d\n", soldQuantity, product.quantity);

} else {

printf("Not enough stock.\n");

}

return 0;

}

O/p: Current Quantity: 50 Enter quantity to sell: 45

Sold 45 units. Updated quantity: 5

**Additional Problem Statements of the structure:**

# Point Distance Calculation:

* + Define a structure for a point in 2D space (x, y).
  + Write a function to calculate the distance between two points. Sol: #include <stdio.h>

#include <math.h>

// Define a structure for a point in 2D space struct Point {

double x; double y;

};

// Function to calculate the distance between two points double calculateDistance(struct Point p1, struct Point p2) {

double distance = sqrt(pow(p2.x - p1.x, 2) + pow(p2.y - p1.y, 2)); return distance;

}

int main() {

struct Point point1, point2;

// Input for the first point

printf("Enter coordinates for point 1 (x y): "); scanf("%lf %lf", &point1.x, &point1.y);

// Input for the second point

printf("Enter coordinates for point 2 (x y): "); scanf("%lf %lf", &point2.x, &point2.y);

// Calculate and display the distance

double distance = calculateDistance(point1, point2); printf("The distance between the points is: %.2f\n", distance);

return 0;

}

O/p: Enter coordinates for point 1 (x y): 5 6 Enter coordinates for point 2 (x y): 4 7

The distance between the points is: 1.41

# Rectangle Properties:

* + Create a structure for a rectangle with length and width.
  + Write functions to calculate the area and perimeter of the rectangle. Sol: #include <stdio.h>

// Define a structure for a rectangle struct Rectangle {

double length; double width;

};

// Function to calculate the area of the rectangle double calculateArea(struct Rectangle rect) {

return rect.length \* rect.width;

}

// Function to calculate the perimeter of the rectangle double calculatePerimeter(struct Rectangle rect) {

return 2 \* (rect.length + rect.width);

}

int main() {

struct Rectangle rect;

// Input for rectangle dimensions printf("Enter the length of the rectangle: "); scanf("%lf", &rect.length);

printf("Enter the width of the rectangle: "); scanf("%lf", &rect.width);

// Calculate and display area and perimeter double area = calculateArea(rect);

double perimeter = calculatePerimeter(rect);

printf("Area of the rectangle: %.2f\n", area); printf("Perimeter of the rectangle: %.2f\n", perimeter);

return 0;

}

O/p:

Enter the length of the rectangle: 12 Enter the width of the rectangle: 32 Area of the rectangle: 384.00

Perimeter of the rectangle: 88.00

# Movie Details:

* + Define a structure to store details of a movie, including title, director, release year, and rating.
  + Write a program to sort movies by their rating. Sol: #include <stdio.h>

#include <string.h>

struct Movie { char title[50];

char director[50]; int releaseYear; double rating;

};

void sortMoviesByRating(struct Movie movies[], int n) { for (int i = 0; i < n - 1; i++) {

for (int j = i + 1; j < n; j++) {

if (movies[i].rating < movies[j].rating) { struct Movie temp = movies[i]; movies[i] = movies[j];

movies[j] = temp;

}

}

}

}

int main() { int n = 3;

struct Movie movies[3] = {

{"Inception", "Christopher Nolan", 2010, 8.8},

{"The Godfather", "Francis Ford Coppola", 1972, 9.2},

{"Interstellar", "Christopher Nolan", 2014, 8.6}

};

sortMoviesByRating(movies, n); printf("Movies sorted by rating:\n"); for (int i = 0; i < n; i++) {

printf("%s (%d), Director: %s, Rating: %.1f\n",

movies[i].title, movies[i].releaseYear, movies[i].director, movies[i].rating);

}

return 0;

}

O/p: Movies sorted by rating:

The Godfather (1972), Director: Francis Ford Coppola, Rating: 9.2 Inception (2010), Director: Christopher Nolan, Rating: 8.8 Interstellar (2014), Director: Christopher Nolan, Rating: 8.6

# Weather Report:

* + Create a structure to store daily weather data, including date, temperature, and humidity.
  + Write a program to find the day with the highest temperature. Sol: #include <stdio.h>

struct Weather {

char date[12]; // Date in format YYYY-MM-DD double temperature;

double humidity;

};

int main() {

int n = 3; // Number of days (can be adjusted) struct Weather data[3] = {

{"2025-01-05", 15.5, 60.0},

{"2025-01-06", 18.2, 55.0},

{"2025-01-07", 21.4, 70.0}

};

int maxIndex = 0;

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

if (data[i].temperature > data[maxIndex].temperature) { maxIndex = i;

}

}

printf("Day with the highest temperature:\n");

printf("Date: %s, Temperature: %.1f, Humidity: %.1f%%\n", data[maxIndex].date, data[maxIndex].temperature,

data[maxIndex].humidity);

return 0;

}

O/p: Day with the highest temperature:

Date: 2025-01-07, Temperature: 21.4, Humidity: 70.0%

# Fraction Arithmetic:

* + Define a structure for a fraction with numerator and denominator.
  + Write functions to add, subtract, multiply, and divide two fractions. Sol: #include <stdio.h>

struct Fraction { int numerator;

int denominator;

};

// Function to find the greatest common divisor (GCD) int gcd(int a, int b) {

while (b != 0) { int temp = b; b = a % b;

a = temp;

}

return a;

}

// Function to simplify a fraction

struct Fraction simplify(struct Fraction frac) {

int divisor = gcd(frac.numerator, frac.denominator); frac.numerator /= divisor;

frac.denominator /= divisor; return frac;

}

// Function to add two fractions

struct Fraction add(struct Fraction f1, struct Fraction f2) { struct Fraction result = {

f1.numerator \* f2.denominator + f2.numerator \* f1.denominator, f1.denominator \* f2.denominator

};

return simplify(result);

}

// Function to subtract two fractions

struct Fraction subtract(struct Fraction f1, struct Fraction f2) { struct Fraction result = {

f1.numerator \* f2.denominator - f2.numerator \* f1.denominator, f1.denominator \* f2.denominator

};

return simplify(result);

}

// Function to multiply two fractions

struct Fraction multiply(struct Fraction f1, struct Fraction f2) {

struct Fraction result = {f1.numerator \* f2.numerator, f1.denominator \* f2.denominator};

return simplify(result);

}

// Function to divide two fractions

struct Fraction divide(struct Fraction f1, struct Fraction f2) {

struct Fraction result = {f1.numerator \* f2.denominator, f1.denominator \* f2.numerator};

return simplify(result);

}

void display(struct Fraction frac) {

printf("%d/%d\n", frac.numerator, frac.denominator);

}

int main() {

struct Fraction f1 = {3, 4};

struct Fraction f2 = {2, 5};

printf("Addition: "); display(add(f1, f2));

printf("Subtraction: ");

display(subtract(f1, f2));

printf("Multiplication: "); display(multiply(f1, f2));

printf("Division: "); display(divide(f1, f2));

return 0;

}

O/p: Addition: 23/20 Subtraction: 7/20 Multiplication: 3/10 Division: 15/8

# Laptop Inventory:

* + Create a structure to represent a laptop with fields for brand, model, processor, RAM, and price.
  + Write a program to list laptops within a specific price range. Sol: #include <stdio.h>

#include <string.h>

struct Laptop { char brand[50];

char model[50]; char processor[50];

int RAM; // in GB double price; // in USD

};

void displayLaptopsInRange(struct Laptop laptops[], int n, double minPrice, double maxPrice) {

printf("Laptops in the price range %.2f to %.2f:\n", minPrice, maxPrice); int found = 0;

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

if (laptops[i].price >= minPrice && laptops[i].price <= maxPrice) { printf("Brand: %s, Model: %s, Processor: %s, RAM: %d GB, Price:

%.2f\n",

laptops[i].brand, laptops[i].model, laptops[i].processor, laptops[i].RAM, laptops[i].price);

found = 1;

}

}

if (!found) {

printf("No laptops found in this price range.\n");

}

}

int main() {

struct Laptop laptops[] = {

{"Dell", "XPS 13", "Intel i7", 16, 1200.00},

{"HP", "Spectre x360", "Intel i5", 8, 900.00},

{"Apple", "MacBook Pro", "M1", 16, 1500.00},

{"Lenovo", "ThinkPad X1", "Intel i7", 32, 1700.00}

};

int n = sizeof(laptops) / sizeof(laptops[0]);

double minPrice, maxPrice; printf("Enter minimum price: "); scanf("%lf", &minPrice); printf("Enter maximum price: "); scanf("%lf", &maxPrice);

displayLaptopsInRange(laptops, n, minPrice, maxPrice);

return 0;

}

O/p: Enter minimum price: 1500

Enter maximum price: 20000

Laptops in the price range 1500.00 to 20000.00:

Brand: Apple, Model: MacBook Pro, Processor: M1, RAM: 16 GB, Price: 1500.00

Brand: Lenovo, Model: ThinkPad X1, Processor: Intel i7, RAM: 32 GB, Price: 1700.00

# Student Attendance:

* + Define a structure to store attendance data, including student ID, total classes, and classes attended.
  + Write a program to calculate and display the attendance percentage for each student.

Sol: #include <stdio.h>

struct Attendance { int studentID; int totalClasses;

int classesAttended;

};

void displayAttendance(struct Attendance students[], int n) { printf("Attendance Percentage for Each Student:\n");

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

double percentage = (students[i].classesAttended / (double)students[i].totalClasses) \* 100;

printf("Student ID: %d, Attendance: %.2f%%\n", students[i].studentID, percentage);

}

}

int main() {

struct Attendance students[] = {

{101, 50, 45},

{102, 60, 50},

{103, 55, 40}

};

int n = sizeof(students) / sizeof(students[0]); displayAttendance(students, n);

return 0;

}

O/p: Attendance Percentage for Each Student:

Student ID: 101, Attendance: 90.00%

Student ID: 102, Attendance: 83.33%

Student ID: 103, Attendance: 72.73%

# Flight Information:

* + Create a structure for a flight with fields for flight number, departure, destination, and duration.
  + Write a program to display flights that are less than a specified duration.

Sol: #include <stdio.h>

#include <string.h>

struct Flight {

char flightNumber[10]; char departure[30]; char destination[30];

double duration; // in hours

};

void displayShortFlights(struct Flight flights[], int n, double maxDuration) { printf("Flights with duration less than %.2f hours:\n", maxDuration);

int found = 0;

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

if (flights[i].duration < maxDuration) {

printf("Flight: %s, Departure: %s, Destination: %s, Duration: %.2f hours\n",

flights[i].flightNumber, flights[i].departure, flights[i].destination, flights[i].duration);

found = 1;

}

}

if (!found) {

printf("No flights found with duration less than %.2f hours.\n", maxDuration);

}

}

int main() {

struct Flight flights[] = {

{"AA101", "New York", "London", 7.5},

{"DL202", "Los Angeles", "Tokyo", 11.0},

{"UA303", "Chicago", "Toronto", 1.5}

};

int n = sizeof(flights) / sizeof(flights[0]);

double maxDuration;

printf("Enter maximum flight duration (hours): "); scanf("%lf", &maxDuration);

displayShortFlights(flights, n, maxDuration);

return 0;

}

O/p: Enter maximum flight duration (hours): 7

Flights with duration less than 7.00 hours:

Flight: UA303, Departure: Chicago, Destination: Toronto, Duration: 1.50 hours

# Polynomial Representation:

* + Define a structure to represent a term of a polynomial (coefficient and exponent).
  + Write functions to add and multiply two polynomials. Sol: #include <stdio.h>

struct Term { int coeff, exp;

};

void addPolynomials(struct Term p1[], int n1, struct Term p2[], int n2) { int i = 0, j = 0;

while (i < n1 && j < n2) {

if (p1[i].exp > p2[j].exp) printf("%dx^%d ", p1[i].coeff, p1[i].exp), i++;

else if (p1[i].exp < p2[j].exp) printf("%dx^%d ", p2[j].coeff, p2[j].exp), j++;

else { printf("%dx^%d ", p1[i].coeff + p2[j].coeff, p1[i].exp); i++; j++; }

}

while (i < n1) printf("%dx^%d ", p1[i].coeff, p1[i].exp), i++; while (j < n2) printf("%dx^%d ", p2[j].coeff, p2[j].exp), j++; printf("\n");

}

void multiplyPolynomials(struct Term p1[], int n1, struct Term p2[], int n2) { for (int i = 0; i < n1; i++) {

for (int j = 0; j < n2; j++) {

printf("%dx^%d ", p1[i].coeff \* p2[j].coeff, p1[i].exp + p2[j].exp);

}

}

printf("\n");

}

int main() {

struct Term p1[] = {{3, 2}, {5, 1}}, p2[] = {{4, 2}, {1, 1}};

printf("Sum: "); addPolynomials(p1, 2, p2, 2); printf("Product: "); multiplyPolynomials(p1, 2, p2, 2);

return 0;

}

O/p: Sum: 7x^2 6x^1

Product: 12x^4 3x^3 20x^3 5x^2

# Medical Records:

* + Create a structure for a patient's medical record with fields for name, age, diagnosis, and treatment.
  + Write a program to search for patients by diagnosis. Sol: #include <stdio.h>

#include <string.h>

struct Patient { char name[50]; int age;

char diagnosis[100]; char treatment[100];

};

void searchByDiagnosis(struct Patient patients[], int n, const char\* diagnosis) { printf("Patients with diagnosis '%s':\n", diagnosis);

int found = 0;

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

if (strstr(patients[i].diagnosis, diagnosis)) {

printf("Name: %s, Age: %d, Treatment: %s\n", patients[i].name, patients[i].age, patients[i].treatment);

found = 1;

}

}

if (!found) printf("No patients found with this diagnosis.\n");

}

int main() {

struct Patient patients[] = {

{"Alice", 30, "Flu", "Rest and fluids"},

{"Bob", 45, "Covid", "Antiviral medication"},

{"Charlie", 60, "Flu", "Antibiotics"},

};

int n = sizeof(patients) / sizeof(patients[0]);

char diagnosis[100];

printf("Enter diagnosis to search: "); scanf("%s", diagnosis);

searchByDiagnosis(patients, n, diagnosis);

return 0;

}

O/p: Enter diagnosis to search: Flu Patients with diagnosis 'Flu':

Name: Alice, Age: 30, Treatment: Rest and fluids Name: Charlie, Age: 60, Treatment: Antibiotics

# Game Scores:

* + Define a structure to store player information, including name, game played, and score.
  + Write a program to display the top scorer for each game. Sol: #include <stdio.h>

#include <string.h>

struct Player { char name[50]; char game[50]; int score;

};

void topScorer(struct Player players[], int n) { char games[10][50]; // Store unique games int gameCount = 0;

// Find unique games

for (int i = 0; i < n; i++) { int found = 0;

for (int j = 0; j < gameCount; j++) {

if (strcmp(players[i].game, games[j]) == 0) { found = 1;

break;

}

}

if (!found) {

strcpy(games[gameCount], players[i].game); gameCount++;

}

}

// Find and display top scorer for each game for (int i = 0; i < gameCount; i++) {

int maxScore = -1; char topPlayer[50];

for (int j = 0; j < n; j++) {

if (strcmp(players[j].game, games[i]) == 0 && players[j].score > maxScore) {

maxScore = players[j].score; strcpy(topPlayer, players[j].name);

}

}

printf("Top scorer for %s: %s with score %d\n", games[i], topPlayer, maxScore);

}

}

int main() {

struct Player players[] = {

{"Alice", "Basketball", 25},

{"Bob", "Basketball", 30},

{"Charlie", "Football", 40},

{"David", "Football", 35}

};

int n = sizeof(players) / sizeof(players[0]);

topScorer(players, n);

return 0;

}

O/p:

Top scorer for Basketball: Bob with score 30 Top scorer for Football: Charlie with score 40

# City Information:

* + Create a structure to store information about a city, including name, population, and area.
  + Write a program to calculate and display the population density of each city.

Sol: #include <stdio.h>

struct City {

char name[50]; int population;

float area; // in square kilometers

};

void displayDensity(struct City cities[], int n) { for (int i = 0; i < n; i++) {

float density = cities[i].population / cities[i].area;

printf("City: %s, Population Density: %.2f people/km²\n", cities[i].name, density);

}

}

int main() {

struct City cities[] = {

{"New York", 8419600, 783.8},

{"Los Angeles", 3980400, 1302},

{"Chicago", 2716000, 589}

};

int n = sizeof(cities) / sizeof(cities[0]);

displayDensity(cities, n);

return 0;

}

O/p: City: New York, Population Density: 10742.03 people/km² City: Los Angeles, Population Density: 3057.14 people/km² City: Chicago, Population Density: 4611.21 people/km²

# Vehicle Registration:

* + Define a structure for vehicle registration details, including registration number, owner, make, and year.
  + Write a program to list all vehicles registered in a given year. Sol: #include <stdio.h>

#include <string.h>

struct Vehicle {

char regNumber[20]; char owner[50];

char make[50];

int year;

};

void listVehiclesByYear(struct Vehicle vehicles[], int n, int year) { printf("Vehicles registered in %d:\n", year);

int found = 0;

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

if (vehicles[i].year == year) {

printf("Reg No: %s, Owner: %s, Make: %s\n", vehicles[i].regNumber, vehicles[i].owner, vehicles[i].make);

found = 1;

}

}

if (!found) printf("No vehicles found for this year.\n");

}

int main() {

struct Vehicle vehicles[] = {

{"ABC123", "John Doe", "Toyota", 2020},

{"XYZ789", "Alice Smith", "Honda", 2021},

{"LMN456", "Bob Johnson", "Ford", 2020}

};

int n = sizeof(vehicles) / sizeof(vehicles[0]);

int year;

printf("Enter year to search for registered vehicles: "); scanf("%d", &year);

listVehiclesByYear(vehicles, n, year);

return 0;

}

O/p: Enter year to search for registered vehicles: 2020 Vehicles registered in 2020:

Reg No: ABC123, Owner: John Doe, Make: Toyota Reg No: LMN456, Owner: Bob Johnson, Make: Ford

# Restaurant Menu:

* + Create a structure to represent a menu item with fields for name, category, and price.
  + Write a program to display menu items in a specific category. Sol: #include <stdio.h>

#include <string.h>

struct MenuItem { char name[50];

char category[50]; float price;

};

void displayItemsByCategory(struct MenuItem menu[], int n, const char\* category) {

printf("Menu items in category '%s':\n", category); int found = 0;

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

if (strcmp(menu[i].category, category) == 0) {

printf("Name: %s, Price: %.2f\n", menu[i].name, menu[i].price); found = 1;

}

}

if (!found) {

printf("No items found in this category.\n");

}

}

int main() {

struct MenuItem menu[] = {

{"Burger", "Fast Food", 5.99},

{"Pizza", "Fast Food", 8.99},

{"Pasta", "Italian", 12.99},

{"Salad", "Vegetarian", 6.49}

};

int n = sizeof(menu) / sizeof(menu[0]);

char category[50];

printf("Enter category to display items: "); scanf("%s", category);

displayItemsByCategory(menu, n, category);

return 0;

}

O/p: Enter category to display items: T Italian Menu items in category 'Italian':

Name: Pasta, Price: 12.99

# Sports Team:

* + Define a structure for a sports team with fields for team name, sport, number of players, and coach.
  + Write a program to display all teams playing a specific sport. Sol: #include <stdio.h>

#include <string.h>

struct SportsTeam { char teamName[50]; char sport[50];

int numPlayers; char coach[50];

};

void displayTeamsBySport(struct SportsTeam teams[], int n, const char\* sport) { printf("Teams playing sport '%s':\n", sport);

int found = 0;

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

if (strcmp(teams[i].sport, sport) == 0) {

printf("Team Name: %s, Players: %d, Coach: %s\n", teams[i].teamName, teams[i].numPlayers, teams[i].coach);

found = 1;

}

}

if (!found) {

printf("No teams found for this sport.\n");

}

}

int main() {

struct SportsTeam teams[] = {

{"Warriors", "Basketball", 12, "Steve Kerr"},

{"Lions", "Football", 11, "John Doe"},

{"Spartans", "Basketball", 12, "Tom Smith"},

{"Eagles", "Football", 11, "Mike Johnson"}

};

int n = sizeof(teams) / sizeof(teams[0]);

char sport[50];

printf("Enter sport to display teams: "); scanf("%s", sport);

displayTeamsBySport(teams, n, sport);

return 0;

}

O/p:

Enter sport to display teams: Football Teams playing sport 'Football':

Team Name: Lions, Players: 11, Coach: John Doe Team Name: Eagles, Players: 11, Coach: Mike Johnson

# Student Marks Analysis:

* + Create a structure to store student marks in different subjects.
  + Write a program to calculate the total and percentage of marks for each student.

Sol: #include <stdio.h>

struct Student { char name[50];

int marks[5]; // Marks in 5 subjects int total;

float percentage;

};

void calculateMarks(struct Student\* student) { student->total = 0;

for (int i = 0; i < 5; i++) {

student->total += student->marks[i];

}

student->percentage = (float)student->total / 5;

}

int main() {

struct Student students[] = {

{"Alice", {85, 90, 78, 92, 88}, 0, 0.0},

{"Bob", {70, 75, 80, 65, 85}, 0, 0.0},

{"Charlie", {90, 85, 95, 80, 89}, 0, 0.0}

};

int n = sizeof(students) / sizeof(students[0]);

for (int i = 0; i < n; i++) { calculateMarks(&students[i]);

printf("Student: %s\nTotal Marks: %d\nPercentage: %.2f%%\n\n", students[i].name, students[i].total, students[i].percentage);

}

return 0;

}

O/p:

Student: Alice Total Marks: 433

Percentage: 86.60%

Student: Bob Total Marks: 375

Percentage: 75.00%

Student: Charlie Total Marks: 439

Percentage: 87.80%

# 17.E-commerce Product:

* + Define a structure for an e-commerce product with fields for product ID, name, category, price, and stock.
  + Write a program to update the stock and calculate the total value of products in stock.

Sol: #include <stdio.h>

struct Product { int productID; char name[50];

char category[50]; float price;

int stock;

};

void updateStock(struct Product\* product, int newStock) { product->stock = newStock;

}

float calculateTotalValue(struct Product product) { return product.price \* product.stock;

}

int main() {

struct Product products[] = {

{101, "Laptop", "Electronics", 799.99, 10},

{102, "Phone", "Electronics", 499.99, 20},

{103, "Shoes", "Footwear", 59.99, 15}

};

int n = sizeof(products) / sizeof(products[0]);

// Display initial stock and total value for (int i = 0; i < n; i++) {

printf("Product: %s, Stock: %d, Total Value: %.2f\n", products[i].name, products[i].stock, calculateTotalValue(products[i]));

}

// Update stock for product 1 (Laptop) updateStock(&products[0], 5); // New stock for Laptop

// Display updated stock and total value printf("\nAfter updating stock:\n");

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

printf("Product: %s, Stock: %d, Total Value: %.2f\n", products[i].name, products[i].stock, calculateTotalValue(products[i]));

}

return 0;

}

O/p: Product: Laptop, Stock: 10, Total Value: 7999.90 Product: Phone, Stock: 20, Total Value: 9999.80 Product: Shoes, Stock: 15, Total Value: 899.85

After updating stock:

Product: Laptop, Stock: 5, Total Value: 3999.95 Product: Phone, Stock: 20, Total Value: 9999.80 Product: Shoes, Stock: 15, Total Value: 899.85

# Music Album:

* + Create a structure to store details of a music album, including album name, artist, genre, and release year.
  + Write a program to display albums of a specific genre. Sol: #include <stdio.h>

#include <string.h>

struct Album {

char albumName[50]; char artist[50];

char genre[50]; int releaseYear;

};

void displayAlbumsByGenre(struct Album albums[], int n, const char\* genre) { printf("Albums of genre '%s':\n", genre);

int found = 0;

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

if (strcmp(albums[i].genre, genre) == 0) {

printf("Album: %s, Artist: %s, Year: %d\n", albums[i].albumName, albums[i].artist, albums[i].releaseYear);

found = 1;

}

}

if (!found) {

printf("No albums found for this genre.\n");

}

}

int main() {

struct Album albums[] = {

{"Thriller", "Michael Jackson", "Pop", 1982},

{"Back in Black", "AC/DC", "Rock", 1980},

{"The Dark Side of the Moon", "Pink Floyd", "Rock", 1973},

{"Future Nostalgia", "Dua Lipa", "Pop", 2020}

};

int n = sizeof(albums) / sizeof(albums[0]);

char genre[50];

printf("Enter genre to display albums: "); scanf("%s", genre);

displayAlbumsByGenre(albums, n, genre);

return 0;

}

O/p: Enter genre to display albums: Rock Albums of genre 'Rock':

Album: Back in Black, Artist: AC/DC, Year: 1980

Album: The Dark Side of the Moon, Artist: Pink Floyd, Year: 1973

# Cinema Ticket Booking:

* + Define a structure for a cinema ticket with fields for movie name, seat number, and price.
  + Write a program to book tickets and display the total revenue generated.

Sol: #include <stdio.h>

struct Ticket {

char movieName[50]; int seatNumber;

float price;

};

float totalRevenue = 0;

void bookTicket(struct Ticket\* ticket, float price) { printf("Enter movie name: ");

getchar(); // to clear the newline from previous input fgets(ticket->movieName, 50, stdin);

ticket->movieName[strcspn(ticket->movieName, "\n")] = 0; // remove newline character

printf("Enter seat number: "); scanf("%d", &ticket->seatNumber); ticket->price = price;

totalRevenue += ticket->price;

printf("Ticket booked for Movie: %s, Seat: %d, Price: %.2f\n", ticket-

>movieName, ticket->seatNumber, ticket->price);

}

int main() {

struct Ticket tickets[5]; // Assume max 5 tickets for simplicity int n = 5;

float price = 12.50; // Price for each ticket

for (int i = 0; i < n; i++) { printf("\nBooking ticket %d\n", i + 1); bookTicket(&tickets[i], price);

}

printf("\nTotal Revenue Generated: %.2f\n", totalRevenue);

return 0;

}

O/p:

Booking ticket 1

Enter movie name: pushpa 2 Enter seat number: 12

Ticket booked for Movie: ushpa 2, Seat: 12, Price: 12.50

Booking ticket 2

Enter movie name: gamechanger Enter seat number: 25

Ticket booked for Movie: gamechanger, Seat: 25, Price: 12.50

Booking ticket 3

Enter movie name: abc Enter seat number: 78

Ticket booked for Movie: abc, Seat: 78, Price: 12.50

Booking ticket 4

Enter movie name: xyz Enter seat number: 56

Ticket booked for Movie: xyz, Seat: 56, Price: 12.50

Booking ticket 5

Enter movie name: ram Enter seat number: 78

Ticket booked for Movie: ram, Seat: 78, Price: 12.50

Total Revenue Generated: 62.50

# University Courses:

* + Create a structure to store course details, including course code, name, instructor, and credits.
  + Write a program to list all courses taught by a specific instructor. Sol: #include <stdio.h>

#include <string.h>

// Structure to represent a course struct Course {

char courseCode[10]; char courseName[100]; char instructor[50];

int credits;

};

// Function to list all courses taught by a specific instructor

void listCoursesByInstructor(struct Course courses[], int numCourses, char instructor[]) {

int found = 0;

printf("Courses taught by %s:\n", instructor);

// Loop through all courses to find the ones taught by the specified instructor for (int i = 0; i < numCourses; i++) {

if (strcmp(courses[i].instructor, instructor) == 0) { printf("Course Code: %s\n", courses[i].courseCode); printf("Course Name: %s\n", courses[i].courseName); printf("Credits: %d\n\n", courses[i].credits);

found = 1; // At least one course found

}

}

// If no courses were found if (!found) {

printf("No courses found for instructor %s.\n", instructor);

}

}

int main() {

// Array of course data for 5 courses struct Course courses[5] = {

{"CS101", "Introduction to Programming", "Dr. Smith", 4},

{"CS102", "Data Structures", "Dr. Smith", 3},

{"MATH101", "Calculus I", "Dr. Johnson", 4},

{"CS103", "Algorithms", "Dr. Smith", 3},

{"PHYS101", "Physics I", "Dr. Williams", 3}

};

char instructor[50];

// Take input for the instructor's name printf("Enter the name of the instructor: "); fgets(instructor, sizeof(instructor), stdin);

instructor[strcspn(instructor, "\n")] = 0; // Remove trailing newline character from input

// List courses taught by the specified instructor

listCoursesByInstructor(courses, 5, instructor);

return 0;

}

O/p: Enter the name of the instructor: Dr. Williams Courses taught by Dr. Williams:

Course Code: PHYS101 Course Name: Physics I Credits: 3