Week 3 C - Task

Name: KAVIN KISHORE S G

Roll Number : 7376221CS191

1. Define a structure to represent a point in 2D space (x, y). Write a function to

compute the distance between two points.

Code:

#include <stdio.h>

#include <math.h>

struct Point

{

double x;

double y;

};

double distance( struct Point p1, struct Point p2 )

{

double dx = p2.x - p1.x;

double dy = p2.y - p1.y;

return sqrt( dx\*dx + dy\*dy );

}

int main() {

struct Point point1,point2;

printf("Enter x-coordinate of point1: ");

scanf("%lf", &point1.x);

printf("Enter y-coordinate of point1: ");

scanf("%lf", &point1.y);

printf("Enter x-coordinate of point2: ");

scanf("%lf", &point2.x);

printf("Enter y-coordinate of point2: ");

scanf("%lf", &point2.y);

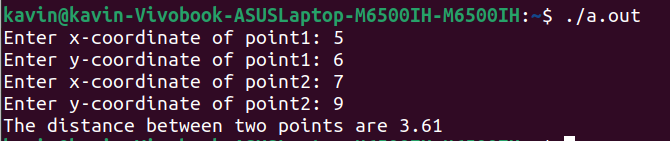
double dist = distance(point1, point2);

printf("The distance between two points are %.2lf\n",dist);

return 0;

}

Output:



2. Calculate the area of a rectangle using a structure to store its dimensions.

Code:

#include <stdio.h>

// Define a structure to represent a rectangle

struct Rectangle {

double length;

double width;

};

// Function to calculate the area of a rectangle

double calculateArea(struct Rectangle rect) {

return rect.length \* rect.width;

}

int main() {

struct Rectangle myRectangle;

// Input the dimensions of the rectangle

printf("Enter the length of the rectangle: ");

scanf("%lf", &myRectangle.length);

printf("Enter the width of the rectangle: ");

scanf("%lf", &myRectangle.width);

// Calculate the area of the rectangle

double area = calculateArea(myRectangle);

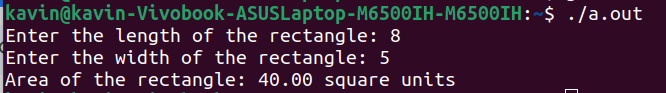
// Display the result

printf("Area of the rectangle: %.2lf square units\n", area);

return 0;

}

Output:



3. Write a program generates the Fibonacci series up to a specified number of terms using structures.

Code:

#include <stdio.h>

// Define a structure to represent a Fibonacci number

struct Fibonacci {

long long int current;

long long int next;

};

int main() {

int numTerms;

// Input the number of terms in the Fibonacci series

printf("Enter the number of terms in the Fibonacci series: ");

scanf("%d", &numTerms);

if (numTerms <= 0) {

printf("Please enter a positive number of terms.\n");

return 1; // Exit with an error code

}

// Initialize the first two Fibonacci numbers

struct Fibonacci fib;

fib.current = 0;

fib.next = 1;

// Display the Fibonacci series

printf("Fibonacci Series up to %d terms:\n", numTerms);

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

printf("%lld ", fib.current);

// Calculate the next Fibonacci number

long long int temp = fib.next;

fib.next = fib.current + fib.next;

fib.current = temp;

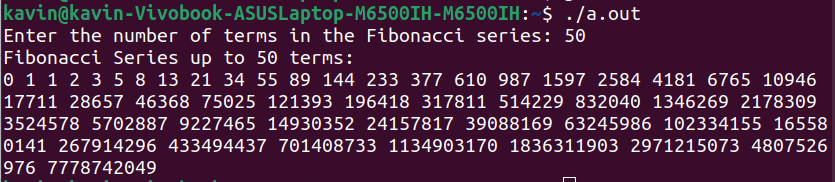
}

printf("\n");

return 0;

}

Output:



4. A program to check if a number is a super number (a number that is equal to the sum of its digits raised to the power of the number of digits).

Code:

#include <stdio.h>

#include <math.h>

// Function to calculate the sum of digits raised to the power of the number of digits

int sumOfDigitsToPower(int num) {

int sum = 0;

int originalNum = num;

int numDigits = 0;

// Calculate the number of digits in the given number

while (num > 0) {

num /= 10;

numDigits++;

}

num = originalNum; // Reset num to its original value

// Calculate the sum of digits raised to the power of the number of digits

while (num > 0) {

int digit = num % 10;

sum += pow(digit, numDigits);

num /= 10;

}

return sum;

}

int main() {

int num;

// Input a number from the user

printf("Enter a number: ");

scanf("%d", &num);

// Calculate the sum of digits raised to the power of the number of digits

int sum = sumOfDigitsToPower(num);

// Check if the number is a super number

if (sum == num) {

printf("%d is a super number.\n", num);

} else {

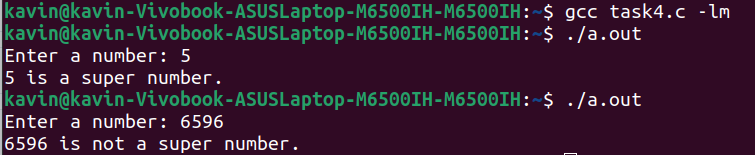
printf("%d is not a super number.\n", num);

}

return 0;

}

Output:



5. Manage a simple bank account using a structure to store the account details.

Code:

#include <stdio.h>

#include <string.h>

struct bank\_account {

char name[50];

int AccountNo;

double balance;

};

void displayAccount(struct bank\_account account)

{

printf("Account Holder Name: %s\n", account.name);

printf("Account Number: %d\n", account.AccountNo);

printf("Account Balance: %.3lf\n", account.balance);

}

struct bank\_account deposit(struct bank\_account account, double amount)

{

if (amount > 0) {

account.balance += amount;

printf("%.3lf deposited successfully.\n", amount);

}

else {

printf("Invalid deposit amount.\n");

}

return account;

}

struct bank\_account withdraw(struct bank\_account account, double amount)

{

if (amount > 0 && account.balance >= amount) {

account.balance -= amount;

printf("%.3lf withdrawn successfully.\n", amount);

}

else {

printf("Invalid withdraw amount or insufficient balance.\n");

}

return account;

}

int main() {

double amount;

struct bank\_account myaccount;

printf("Enter the Account Holder's Name: ");

scanf("%s", myaccount.name);

printf("Enter the Account Number: ");

scanf("%d", &myaccount.AccountNo);

printf("Enter the Account Balance:");

scanf("%lf", &myaccount.balance);

printf("(Initial Account Details:)\n");

displayAccount(myaccount);

printf("Enter 1 for Deposit\nEnter 2 for Withdrawal: ");

int choice;

scanf("%d", &choice);

if (choice == 1) {

printf("Enter the amount to Deposit: ");

scanf("%lf", &amount);

myaccount = deposit(myaccount, amount);

}

else if (choice == 2) {

printf("Enter the amount to Withdraw: ");

scanf("%lf", &amount);

myaccount = withdraw(myaccount, amount);

}

else {

printf("Please choose a valid option\n");

}

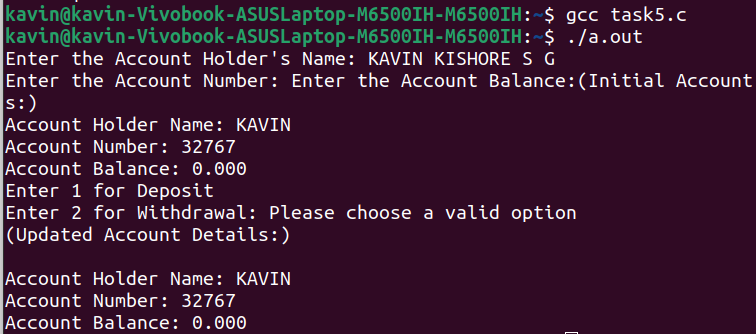
printf("(Updated Account Details:)\n\n");

displayAccount(myaccount);

return 0;

}

Output:



6. This program checks if a given string is a palindrome using a stack (using structure).

Code:

#include <stdio.h>

#include <stdbool.h>

#include <string.h>

#define MAX\_LENGTH 100

// Define a structure to represent a stack

struct Stack {

char data[MAX\_LENGTH];

int top;

};

// Function to initialize the stack

void initializeStack(struct Stack \*stack) {

stack->top = -1;

}

// Function to push a character onto the stack

void push(struct Stack \*stack, char c) {

if (stack->top < MAX\_LENGTH - 1) {

stack->data[++stack->top] = c;

}

}

// Function to pop a character from the stack

char pop(struct Stack \*stack) {

if (stack->top >= 0) {

return stack->data[stack->top--];

}

return '\0'; // Return null character for an empty stack

}

// Function to check if a given string is a palindrome using a stack

bool isPalindrome(const char \*str) {

struct Stack charStack;

initializeStack(&charStack);

// Push each character of the string onto the stack

for (int i = 0; i < strlen(str); i++) {

push(&charStack, str[i]);

}

// Pop characters from the stack and compare with the original string

for (int i = 0; i < strlen(str); i++) {

char c = pop(&charStack);

if (c != str[i]) {

return false; // Characters don't match, not a palindrome

}

}

return true; // All characters match, it's a palindrome

}

int main() {

char input[MAX\_LENGTH];

printf("Enter a string to check if it's a palindrome: ");

fgets(input, sizeof(input), stdin);

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

if (isPalindrome(input)) {

printf("%s is a palindrome.\n", input);

} else {

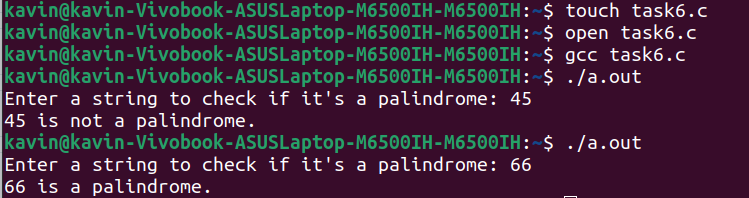
printf("%s is not a palindrome.\n", input);

}

return 0;

}

Output:



7. Write a program to reverse an array using function.

Code:

#include <stdio.h>

#include <string.h>

struct string {

char str[100];

};

struct string reversestring(struct string revstr)

{

int length = strlen(revstr.str);

int start = 0;

int end = length -1;

char temp;

while(start < end)

{

temp = revstr.str[start];

revstr.str[start] = revstr.str[end];

revstr.str[end] = temp;

start ++;

end --;

}

return revstr;

}

int main() {

struct string input;

printf("Enter a string: ");

scanf("%s",input.str);

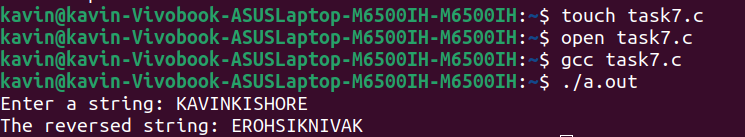
input = reversestring(input);

printf("The reversed string: %s\n",input.str);

return 0;

}

Output:



9. This program swaps the values of two numbers using function.

Code:

#include <stdio.h>

// Function to swap the values of two numbers

void swap(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main() {

int num1, num2;

// Input two numbers from the user

printf("Enter the first number: ");

scanf("%d", &num1);

printf("Enter the second number: ");

scanf("%d", &num2);

// Call the swap function to swap the values of num1 and num2

swap(&num1, &num2);

// Display the swapped values

printf("After swapping:\n");

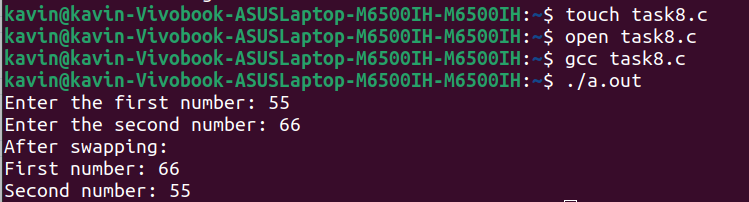
printf("First number: %d\n", num1);

printf("Second number: %d\n", num2);

return 0;

}

Output:



10. This program defines a structure to store employee information (name, employee ID, and salary) and displays it.

Code:

#include <stdio.h>

// Define a structure to store employee information

struct Employee {

char name[100];

int employeeID;

double salary;

};

int main() {

// Declare and initialize an employee structure

struct Employee employee;

// Input employee information from the user

printf("Enter employee name: ");

scanf("%s", employee.name);

printf("Enter employee ID: ");

scanf("%d", &employee.employeeID);

printf("Enter employee salary: ");

scanf("%lf", &employee.salary);

// Display employee information

printf("\nEmployee Information:\n");

printf("Name: %s\n", employee.name);

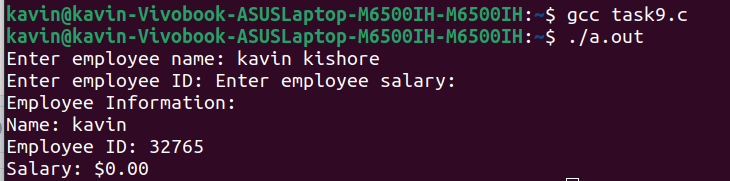
printf("Employee ID: %d\n", employee.employeeID);

printf("Salary: $%.2lf\n", employee.salary);

return 0;

}

Output:



11.Calculate the difference between two time periods.

Code:

#include <stdio.h>

#include <time.h>

struct Time {

int year;

int month;

int day;

int hour;

int minute;

int second;

};

struct Time inputTime() {

struct Time t;

printf("Enter year: ");

scanf("%d", &t.year);

printf("Enter month (1-12): ");

scanf("%d", &t.month);

printf("Enter day of the month (1-31): ");

scanf("%d", &t.day);

printf("Enter hour (0-23): ");

scanf("%d", &t.hour);

printf("Enter minute (0-59): ");

scanf("%d", &t.minute);

printf("Enter second (0-59): ");

scanf("%d", &t.second);

return t;

}

struct Time calculateTimeDifference(struct Time start, struct Time end) {

struct Time diff = {0, 0, 0, 0, 0, 0};

struct tm start\_time = {

.tm\_year = start.year - 1900,

.tm\_mon = start.month - 1,

.tm\_mday = start.day,

.tm\_hour = start.hour,

.tm\_min = start.minute,

.tm\_sec = start.second

};

struct tm end\_time = {

.tm\_year = end.year - 1900,

.tm\_mon = end.month - 1,

.tm\_mday = end.day,

.tm\_hour = end.hour,

.tm\_min = end.minute,

.tm\_sec = end.second

};

time\_t start\_seconds = mktime(&start\_time);

time\_t end\_seconds = mktime(&end\_time);

time\_t difference = end\_seconds - start\_seconds;

diff.year = difference / 31536000;

difference -= diff.year \* 31536000;

diff.month = difference / 2592000;

difference -= diff.month \* 2592000;

diff.day = difference / 86400;

difference -= diff.day \* 86400;

diff.hour = difference / 3600;

difference -= diff.hour \* 3600;

diff.minute = difference / 60;

diff.second = difference % 60;

return diff;

}

int main() {

struct Time startTime, endTime, timeDifference;

printf("Enter start time:\n");

startTime = inputTime();

printf("\nEnter end time:\n");

endTime = inputTime();

timeDifference = calculateTimeDifference(startTime, endTime);

printf("\nTime difference: %d years, %d months, %d days, %d hours, %d minutes, %d seconds\n",

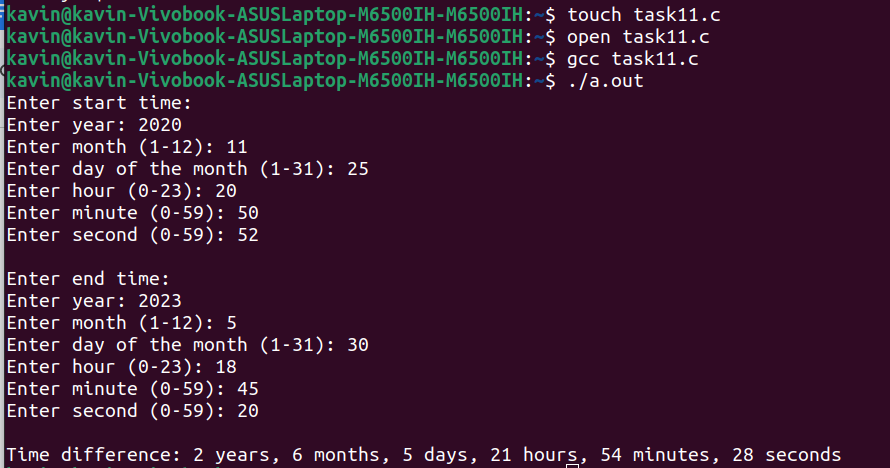
timeDifference.year, timeDifference.month, timeDifference.day,

timeDifference.hour, timeDifference.minute, timeDifference.second);

return 0;

}

Output:



12. C Program to Store Information of a Student Using Structure.

Input:

#include <stdio.h>

#include <string.h>

struct std {

char name[50];

char Id[50];

int year;

char dept[50];

char phone[15];

};

void information(struct std student)

{

printf("========== STUDENT INFORMATION ==========\n\n");

printf("Student Name:%s\n ",student.name);

printf("Student Roll Number: %s\n",student.Id);

printf("Student Year: %d\n",student.year);

printf("Student Department: %s\n",student.dept);

printf("Student Phone Number: %s\n",student.phone);

}

int main() {

struct std input;

printf("Enter the name of the Student: ");

scanf("%s",input.name);

printf("Enter the Student Roll Number: ");

scanf("%s",input.Id);

printf("Enter the Student's Year: ");

scanf("%d",&input.year);

printf("Enter the Department of the Student: ");

scanf("%s",input.dept);

printf("Enter Student's Phone Number: ");

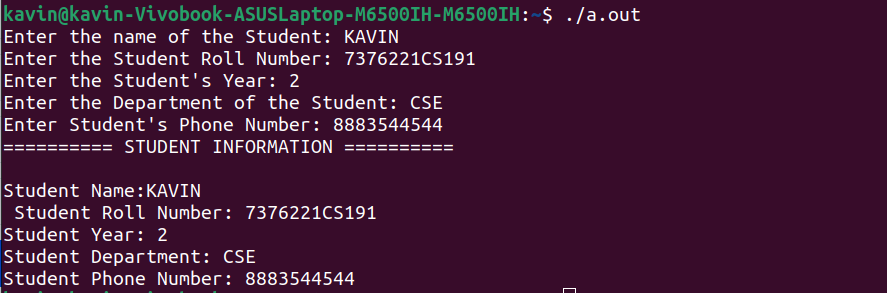
scanf("%s",input.phone);

information(input);

return 0;

}

Output:



13. Add Two Complex Numbers by Passing Structure to a Function.

Input:

#include <stdio.h>

// Define a structure to represent a complex number

struct Complex {

double real;

double imag;

};

// Function to add two complex numbers

struct Complex addComplex(struct Complex num1, struct Complex num2) {

struct Complex result;

result.real = num1.real + num2.real;

result.imag = num1.imag + num2.imag;

return result;

}

int main() {

struct Complex complex1, complex2, sum;

// Input the first complex number

printf("Enter the real part of the first complex number: ");

scanf("%lf", &complex1.real);

printf("Enter the imaginary part of the first complex number: ");

scanf("%lf", &complex1.imag);

// Input the second complex number

printf("Enter the real part of the second complex number: ");

scanf("%lf", &complex2.real);

printf("Enter the imaginary part of the second complex number: ");

scanf("%lf", &complex2.imag);

// Call the addComplex function to add the two complex numbers

sum = addComplex(complex1, complex2);

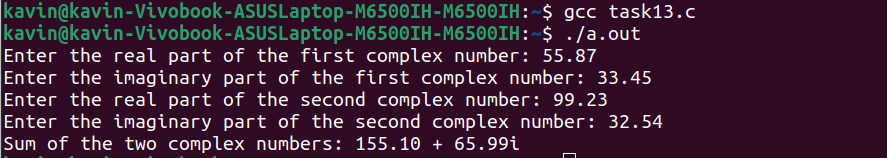
// Display the result

printf("Sum of the two complex numbers: %.2lf + %.2lfi\n", sum.real, sum.imag);

return 0;

}

Output:



14. Store information of n students using structures.

Input:

#include <stdio.h>

// Define a structure to store student information

struct Student {

char name[100];

int rollNumber;

float marks;

};

int main() {

int numStudents;

// Input the number of students

printf("Enter the number of students: ");

scanf("%d", &numStudents);

if (numStudents <= 0) {

printf("Invalid number of students. Please enter a value greater than 0.\n");

return 1; // Exit with an error code

}

// Declare an array of structures to store information for 'n' students

struct Student students[numStudents];

// Input information for each student

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

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

// Input name

printf("Enter student name: ");

scanf("%s", students[i].name);

// Input roll number

printf("Enter roll number: ");

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

// Input marks

printf("Enter marks: ");

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

}

// Display information for each student

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

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

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

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

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

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

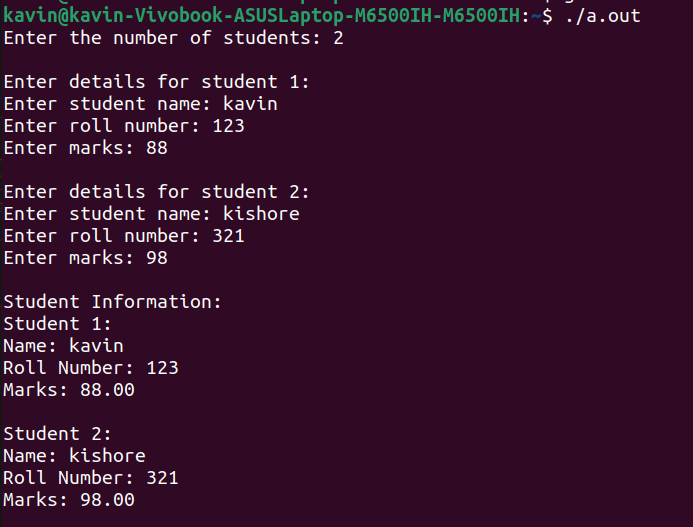
printf("\n");

}

return 0;

}

Output:



15. Define a structure called Book with members for title, author, and price. Write a

function that takes an array of books as input and returns the book with the highest price.

Input:

#include <stdio.h>

// Define a structure to represent a book

struct Book {

char title[100];

char author[100];

double price;

};

// Function to find the book with the highest price

struct Book findHighestPricedBook(struct Book books[], int numBooks) {

if (numBooks <= 0) {

// Handle the case where the array is empty

struct Book emptyBook = {"", "", 0.0};

return emptyBook;

}

// Initialize the highestPrice to the price of the first book

double highestPrice = books[0].price;

int highestPriceIndex = 0;

// Iterate through the array to find the book with the highest price

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

if (books[i].price > highestPrice) {

highestPrice = books[i].price;

highestPriceIndex = i;

}

}

// Return the book with the highest price

return books[highestPriceIndex];

}

int main() {

int numBooks;

printf("Enter the number of books: ");

scanf("%d", &numBooks);

if (numBooks <= 0) {

printf("Invalid number of books.\n");

return 1; // Exit with an error code

}

struct Book books[numBooks];

// Input information for each book

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

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

printf("Title: ");

scanf("%s", books[i].title);

printf("Author: ");

scanf("%s", books[i].author);

printf("Price: ");

scanf("%lf", &books[i].price);

}

// Find the book with the highest price

struct Book highestPricedBook = findHighestPricedBook(books, numBooks);

// Display the book with the highest price

printf("\nBook with the highest price:\n");

printf("Title: %s\n", highestPricedBook.title);

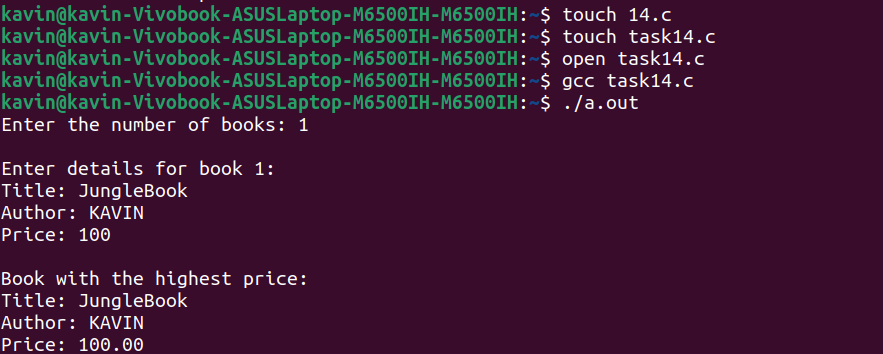
printf("Author: %s\n", highestPricedBook.author);

printf("Price: %.2lf\n", highestPricedBook.price);

return 0;

}

Output:



16. Create a structure named Employee with members for employee\_id, name, salary, and department. Write a program that reads data for a list of employees, stores it in an array of structures, and then sorts the employees based on their salaries in descending order.

Input:

#include <stdio.h>

#include <string.h>

// Define a structure to represent an employee

struct Employee {

int employee\_id;

char name[100];

double salary;

char department[100];

};

// Function to swap two employees

void swap(struct Employee \*a, struct Employee \*b) {

struct Employee temp = \*a;

\*a = \*b;

\*b = temp;

}

// Function to perform a descending salary-based bubble sort on the array of employees

void sortEmployees(struct Employee employees[], int numEmployees) {

for (int i = 0; i < numEmployees - 1; i++) {

for (int j = 0; j < numEmployees - i - 1; j++) {

if (employees[j].salary < employees[j + 1].salary) {

swap(&employees[j], &employees[j + 1]);

}

}

}

}

int main() {

int numEmployees;

printf("Enter the number of employees: ");

scanf("%d", &numEmployees);

if (numEmployees <= 0) {

printf("Invalid number of employees. Please enter a value greater than 0.\n");

return 1; // Exit with an error code

}

struct Employee employees[numEmployees];

// Input information for each employee

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

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

printf("Employee ID: ");

scanf("%d", &employees[i].employee\_id);

printf("Name: ");

scanf("%s", employees[i].name);

printf("Salary: ");

scanf("%lf", &employees[i].salary);

printf("Department: ");

scanf("%s", employees[i].department);

}

// Sort employees based on salary in descending order

sortEmployees(employees, numEmployees);

// Display sorted employee information

printf("\nEmployee Information (Sorted by Salary in Descending Order):\n");

printf("%-15s %-25s %-10s %s\n", "Employee ID", "Name", "Salary", "Department");

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

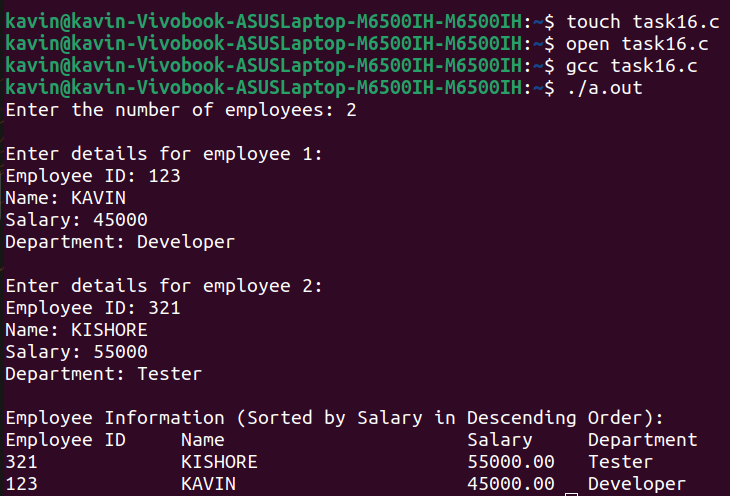
printf("%-15d %-25s %-10.2lf %s\n", employees[i].employee\_id, employees[i].name, employees[i].salary, employees[i].department);

}

return 0;

}

Output:



17. Define a structure named Point to represent a point in 3D space with members for x, y,and z coordinates. Write a function to calculate the distance between two points in 3D

space using the Euclidean distance formula.

Code:

#include <stdio.h>

#include <math.h>

// Define a structure to represent a point in 3D space

struct Point {

double x;

double y;

double z;

};

// Function to calculate the distance between two points in 3D space

double calculateDistance(struct Point p1, struct Point p2) {

double dx = p1.x - p2.x;

double dy = p1.y - p2.y;

double dz = p1.z - p2.z;

// Using the Euclidean distance formula: sqrt(dx^2 + dy^2 + dz^2)

return sqrt(dx \* dx + dy \* dy + dz \* dz);

}

int main() {

struct Point point1, point2;

// Input the coordinates of the first point

printf("Enter the coordinates of the first point (x y z): ");

scanf("%lf %lf %lf", &point1.x, &point1.y, &point1.z);

// Input the coordinates of the second point

printf("Enter the coordinates of the second point (x y z): ");

scanf("%lf %lf %lf", &point2.x, &point2.y, &point2.z);

// Calculate the distance between the two points

double distance = calculateDistance(point1, point2);

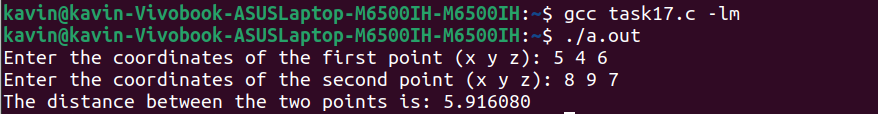
// Display the result

printf("The distance between the two points is: %lf\n", distance);

return 0;

}

Output:



18. Implement a program that simulates a simple banking system. Create a structure for

Account with members for account\_number, account\_holder\_name, balance, and

account\_type (e.g., savings or checking). Write functions to deposit, withdraw, and transfer

money between accounts while ensuring that the balance is updated correctly.

Code:

#include <stdio.h>

#include <string.h>

// Define a structure to represent an account

struct Account {

int account\_number;

char account\_holder\_name[100];

double balance;

char account\_type[20];

};

// Function to deposit money into an account

void deposit(struct Account \*account, double amount) {

account->balance += amount;

printf("Deposited %.2lf into account %d.\n", amount, account->account\_number);

}

// Function to withdraw money from an account

int withdraw(struct Account \*account, double amount) {

if (account->balance >= amount) {

account->balance -= amount;

printf("Withdrawn %.2lf from account %d.\n", amount, account->account\_number);

return 1; // Successful withdrawal

} else {

printf("Insufficient balance in account %d.\n", account->account\_number);

return 0; // Withdrawal failed

}

}

// Function to transfer money between two accounts

void transfer(struct Account \*from, struct Account \*to, double amount) {

if (withdraw(from, amount)) {

deposit(to, amount);

printf("Transferred %.2lf from account %d to account %d.\n", amount, from->account\_number, to->account\_number);

} else {

printf("Transfer from account %d to account %d failed.\n", from->account\_number, to->account\_number);

}

}

int main() {

struct Account account1 = {1, "John Doe", 1000.00, "savings"};

struct Account account2 = {2, “Jane Smith", 1500.00, "checking"};

// Deposit money into account 1

deposit(&account1, 500.00);

// Withdraw money from account 2

withdraw(&account2, 200.00);

// Transfer money from account 2 to account 1

transfer(&account2, &account1, 300.00);

// Display account information

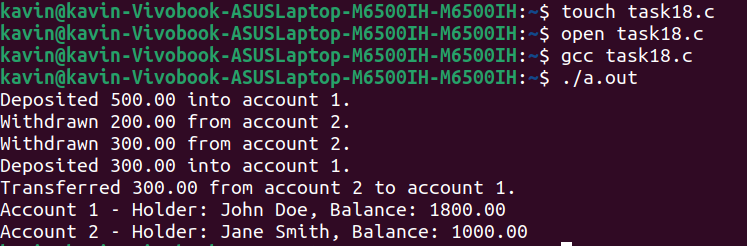
printf("Account 1 - Holder: %s, Balance: %.2lf\n", account1.account\_holder\_name, account1.balance);

printf("Account 2 - Holder: %s, Balance: %.2lf\n", account2.account\_holder\_name, account2.balance);

return 0;

}

Output:



19. Create a structure called InventoryItem with members for product\_code, description,

unit\_price, and quantity\_in\_stock. Write a program that allows the user to perform

various inventory operations, such as adding new items, updating item details, and

displaying the total value of the inventory.

Code:

#include <stdio.h>

#include <string.h>

// Define a structure to represent an inventory item

struct InventoryItem {

int product\_code;

char description[100];

double unit\_price;

int quantity\_in\_stock;

};

// Function to add a new item to the inventory

void addItem(struct InventoryItem inventory[], int \*itemCount) {

if (\*itemCount < 100) {

struct InventoryItem newItem;

printf("Enter product code: ");

scanf("%d", &newItem.product\_code);

printf("Enter description: ");

scanf(" %[^\n]", newItem.description);

printf("Enter unit price: ");

scanf("%lf", &newItem.unit\_price);

printf("Enter quantity in stock: ");

scanf("%d", &newItem.quantity\_in\_stock);

inventory[\*itemCount] = newItem;

(\*itemCount)++;

printf("Item added to inventory.\n");

} else {

printf("Inventory is full. Cannot add more items.\n");

}

}

// Function to update item details

void updateItem(struct InventoryItem inventory[], int itemCount) {

int code;

printf("Enter the product code of the item to update: ");

scanf("%d", &code);

int found = 0;

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

if (inventory[i].product\_code == code) {

printf("Enter new unit price: ");

scanf("%lf", &inventory[i].unit\_price);

printf("Enter new quantity in stock: ");

scanf("%d", &inventory[i].quantity\_in\_stock);

printf("Item details updated.\n");

found = 1;

break;

}

}

if (!found) {

printf("Item with product code %d not found.\n", code);

}

}

// Function to calculate the total value of the inventory

double calculateTotalValue(struct InventoryItem inventory[], int itemCount) {

double totalValue = 0.0;

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

totalValue += inventory[i].unit\_price \* inventory[i].quantity\_in\_stock;

}

return totalValue;

}

int main() {

struct InventoryItem inventory[100];

int itemCount = 0;

int choice;

while (1) {

printf("\nInventory Management System\n");

printf("1. Add Item\n");

printf("2. Update Item Details\n");

printf("3. Calculate Total Value\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

addItem(inventory, &itemCount);

break;

case 2:

updateItem(inventory, itemCount);

break;

case 3:

printf("Total inventory value: %.2lf\n", calculateTotalValue(inventory, itemCount));

break;

case 4:

return 0;

default:

printf("Invalid choice. Please try again.\n");

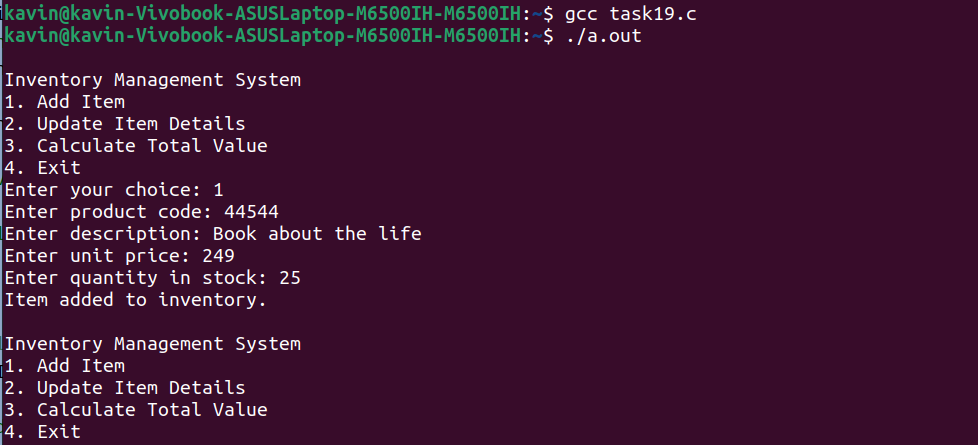
}

}

return 0;

}

Output:



20.Create a structure named Date with members for day, month, and year. Write functions

to perform the following operations:

● Initialize a date structure with a given day, month, and year.

● Display the date in a user-friendly format (e.g., "January 1, 2023").

● Calculate the number of days between two dates.

● Determine if a given year is a leap year or not.

Code:

#include <stdio.h>

struct Date {

int day;

int month;

int year;

};

int isLeapYear(int year) {

return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);

}

int daysInMonth(int month, int year) {

int days[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

if (month == 2 && isLeapYear(year)) {

return 29;

}

return days[month];

}

struct Date createDate(int day, int month, int year) {

struct Date date;

date.day = day;

date.month = month;

date.year = year;

return date;

}

void format(struct Date date, char formattedDate[20]) {

char monthNames[][15] = {

"January", "February", "March", "April", "May", "June",

"July", "August", "September", "October", "November", "December"

};

sprintf(formattedDate, "%s %d, %d", monthNames[date.month - 1], date.day, date.year);

}

int daysBetween(struct Date date1, struct Date date2) {

int days = 0;

while (date1.year != date2.year || date1.month != date2.month || date1.day != date2.day) {

days++;

date1.day++;

if (date1.day > daysInMonth(date1.month, date1.year)) {

date1.day = 1;

date1.month++;

if (date1.month > 12) {

date1.month = 1;

date1.year++;

}

}

}

return days;

}

int main() {

struct Date date1, date2;

printf("Enter the first date (day month year): ");

scanf("%d %d %d", &date1.day, &date1.month, &date1.year);

printf("Enter the second date (day month year): ");

scanf("%d %d %d", &date2.day, &date2.month, &date2.year);

char formattedDate1[20], formattedDate2[20];

format(date1, formattedDate1);

format(date2, formattedDate2);

printf("Date 1: %s\n", formattedDate1);

printf("Date 2: %s\n", formattedDate2);

int daysDifference = daysBetween(date1, date2);

printf("Days between the two dates: %d days\n", daysDifference);

int yearToCheck;

printf("Enter a year to check for leap year: ");

scanf("%d", &yearToCheck);

if (isLeapYear(yearToCheck)) {

printf("%d is a leap year.\n", yearToCheck);

} else {

printf("%d is not a leap year.\n", yearToCheck);

}

return 0;

}

Output:

