1. #include <stdio.h>

#define MAX\_ROWS 100

#define MAX\_COLS 100

// Function to add two matrices

void addMatrices(int rows, int cols, int matrix1[MAX\_ROWS][MAX\_COLS], int matrix2[MAX\_ROWS][MAX\_COLS], int result[MAX\_ROWS][MAX\_COLS]) {

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

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

result[i][j] = matrix1[i][j] + matrix2[i][j];

}

}

}

// Function to display a matrix

void displayMatrix(int rows, int cols, int matrix[MAX\_ROWS][MAX\_COLS]) {

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

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

printf("%d\t", matrix[i][j]);

}

printf("\n");

}

}

int main() {

int rows, cols;

// Input matrix dimensions

printf("Enter the number of rows and columns for the matrices: ");

scanf("%d %d", &rows, &cols);

// Input matrices

printf("Enter elements for matrix1:\n");

int matrix1[MAX\_ROWS][MAX\_COLS];

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

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

scanf("%d", &matrix1[i][j]);

}

}

printf("Enter elements for matrix2:\n");

int matrix2[MAX\_ROWS][MAX\_COLS];

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

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

scanf("%d", &matrix2[i][j]);

}

}

// Matrix addition

int result[MAX\_ROWS][MAX\_COLS];

addMatrices(rows, cols, matrix1, matrix2, result);

// Display the result

printf("Resultant Matrix after addition:\n");

displayMatrix(rows, cols, result);

return 0;

}

1. #include <stdio.h>

// Function to perform matrix multiplication

void multiplyMatrix(int firstMatrix[10][10], int secondMatrix[10][10], int result[10][10], int rowFirst, int columnFirst, int rowSecond, int columnSecond) {

// Initializing elements of result matrix to 0

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

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

result[i][j] = 0;

}

}

// Multiplying firstMatrix and secondMatrix and storing result in result matrix

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

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

for (int k = 0; k < columnFirst; ++k) {

result[i][j] += firstMatrix[i][k] \* secondMatrix[k][j];

}

}

}

}

// Function to display a matrix

void displayMatrix(int matrix[10][10], int row, int column) {

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

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

printf("%d\t", matrix[i][j]);

}

printf("\n");

}

}

int main() {

int firstMatrix[10][10], secondMatrix[10][10], result[10][10];

int rowFirst, columnFirst, rowSecond, columnSecond;

printf("Enter rows and columns for first matrix: ");

scanf("%d %d", &rowFirst, &columnFirst);

printf("Enter elements of matrix 1:\n");

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

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

scanf("%d", &firstMatrix[i][j]);

}

}

printf("Enter rows and columns for second matrix: ");

scanf("%d %d", &rowSecond, &columnSecond);

// Checking if multiplication is possible

if (columnFirst != rowSecond) {

printf("Multiplication is not possible. Column of the first matrix should be equal to the row of the second matrix.\n");

} else {

printf("Enter elements of matrix 2:\n");

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

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

scanf("%d", &secondMatrix[i][j]);

}

}

// Calling the function to perform matrix multiplication

multiplyMatrix(firstMatrix, secondMatrix, result, rowFirst

1. #include <stdio.h>

// Function to reverse a string

void reverseString(char str[]) {

int length = 0;

// Calculate the length of the string

while (str[length] != '\0') {

length++;

}

// Reverse the string

for (int i = 0; i < length / 2; i++) {

char temp = str[i];

str[i] = str[length - i - 1];

str[length - i - 1] = temp;

}

}

int main() {

char inputString[100];

printf("Enter a string: ");

gets(inputString); // Note: gets() is used for simplicity; in a real program, consider using fgets() for better security.

// Call the function to reverse the string

reverseString(inputString);

// Display the reversed string

printf("Reversed string: %s\n", inputString);

return 0;

}

a)#include <stdio.h>

#include <string.h>

// Function to check if a string is a palindrome

int isPalindrome(char str[]) {

int length = strlen(str);

for (int i = 0; i < length / 2; i++) {

if (str[i] != str[length - i - 1]) {

return 0; // Not a palindrome

}

}

return 1; // Palindrome

}

int main() {

char inputString[100];

printf("Enter a string: ");

gets(inputString); // Note: gets() is used for simplicity; in a real program, consider using fgets().

if (isPalindrome(inputString)) {

printf("The string is a palindrome.\n");

} else {

printf("The string is not a palindrome.\n");

}

return 0;

}

#include <stdio.h>

#include <string.h>

int main() {

char inputString[100];

printf("Enter a string: ");

gets(inputString); // Note: gets() is used for simplicity; in a real program, consider using fgets().

int length = strlen(inputString);

printf("Length of the string: %d\n", length);

return 0;

}

b) #include <stdio.h>

#include <string.h>

int main() {

char inputString[100];

printf("Enter a string: ");

gets(inputString); // Note: gets() is used for simplicity; in a real program, consider using fgets().

int length = strlen(inputString);

printf("Length of the string: %d\n", length);

return 0;

}

c) #include <stdio.h>

#include <string.h>

int main() {

char firstString[100], secondString[100];

printf("Enter the first string: ");

gets(firstString); // Note: gets() is used for simplicity; in a real program, consider using fgets().

printf("Enter the second string: ");

gets(secondString);

// Using strcat() to concatenate the strings

strcat(firstString, secondString);

printf("Concatenated string: %s\n", firstString);

return 0;

}

1. #include <stdio.h>

// Function to calculate factorial of a number

int factorial(int num) {

if (num == 0 || num == 1) {

return 1;

} else {

return num \* factorial(num - 1);

}

}

// Function to calculate nCr

int nCr(int n, int r) {

return factorial(n) / (factorial(r) \* factorial(n - r));

}

int main() {

int n, r;

printf("Enter the value of n: ");

scanf("%d", &n);

printf("Enter the value of r: ");

scanf("%d", &r);

// Checking if n is greater than or equal to r

if (n < r) {

printf("Invalid input. n should be greater than or equal to r.\n");

} else {

int result = nCr(n, r);

printf("%dC%d is: %d\n", n, r, result);

}

return 0;

}

1. #include <stdio.h>

// Recursive function to generate Fibonacci series

int fibonacci(int n) {

if (n <= 1) {

return n;

} else {

return fibonacci(n - 1) + fibonacci(n - 2);

}

}

// Function to print Fibonacci series up to n terms

void printFibonacciSeries(int n) {

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

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

printf("%d ", fibonacci(i));

}

printf("\n");

}

int main() {

int terms;

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

scanf("%d", &terms);

if (terms <= 0) {

printf("Invalid input. Number of terms should be greater than 0.\n");

} else {

// Call the function to print the Fibonacci series

printFibonacciSeries(terms);

}

return 0;

}

7.

a) #include <stdio.h>

int main() {

FILE \*file;

char data[100];

// Open the file in write mode

file = fopen("example.txt", "w");

if (file == NULL) {

printf("Error opening the file.\n");

return 1;

}

// Input data to write into the file

printf("Enter data to write into the file:\n");

gets(data);

// Write data into the file

fprintf(file, "%s", data);

// Close the file

fclose(file);

printf("Data written to the file successfully.\n");

return 0;

}

b) #include <stdio.h>

int main() {

FILE \*file;

char data[100];

// Open the file in read mode

file = fopen("example.txt", "r");

if (file == NULL) {

printf("Error opening the file.\n");

return 1;

}

// Read data from the file

fscanf(file, "%[^\n]", data);

// Close the file

fclose(file);

// Display the read data

printf("Data read from the file:\n%s\n", data);

return 0;

}

c) #include <stdio.h>

int main() {

FILE \*file;

char data[100];

// Open the file in append mode

file = fopen("example.txt", "a");

if (file == NULL) {

printf("Error opening the file.\n");

return 1;

}

// Input data to append to the file

printf("Enter data to append to the file:\n");

gets(data);

// Append data to the file

fprintf(file, "%s", data);

// Close the file

fclose(file);

printf("Data appended to the file successfully.\n");

return 0;

}

8) #include <stdio.h>

#include <math.h>

// Function to perform addition

double add(double a, double b) {

return a + b;

}

// Function to perform subtraction

double subtract(double a, double b) {

return a - b;

}

// Function to perform multiplication

double multiply(double a, double b) {

return a \* b;

}

// Function to perform division

double divide(double a, double b) {

if (b != 0) {

return a / b;

} else {

printf("Error: Cannot divide by zero.\n");

return 0;

}

}

// Function to perform exponentiation

double exponent(double base, double exponent) {

return pow(base, exponent);

}

int main() {

int choice;

double num1, num2;

do {

// Display menu

printf("\nMath Operations Menu:\n");

printf("1. Addition\n");

printf("2. Subtraction\n");

printf("3. Multiplication\n");

printf("4. Division\n");

printf("5. Exponentiation\n");

printf("0. Exit\n");

// Get user choice

printf("Enter your choice (0-5): ");

scanf("%d", &choice);

switch (choice) {

case 1:

// Addition

printf("Enter two numbers: ");

scanf("%lf %lf", &num1, &num2);

printf("Result: %.2lf\n", add(num1, num2));

break;

case 2:

// Subtraction

printf("Enter two numbers: ");

scanf("%lf %lf", &num1, &num2);

printf("Result: %.2lf\n", subtract(num1, num2));

break;

case 3:

// Multiplication

printf("Enter two numbers: ");

scanf("%lf %lf", &num1, &num2);

printf("Result: %.2lf\n", multiply(num1, num2));

break;

case 4:

// Division

printf("Enter two numbers: ");

scanf("%lf %lf", &num1, &num2);

printf("Result: %.2lf\n", divide(num1, num2));

break;

case 5:

// Exponentiation

printf("Enter base and exponent: ");

scanf("%lf %lf", &num1, &num2);

printf("Result: %.2lf\n", exponent(num1, num2));

break;

case 0:

// Exit

printf("Exiting the program.\n");

break;

default:

// Invalid choice

printf("Invalid choice. Please enter a number between 0 and 5.\n");

}

} while (choice != 0);

return 0;

}

9) #include <stdio.h>

// Function to perform linear search in an array

int linearSearch(int array[], int size, int key) {

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

if (array[i] == key) {

return i; // Return the index where the key is found

}

}

return -1; // Return -1 if the key is not found

}

int main() {

int array[100], size, key;

// Input the size of the array

printf("Enter the size of the array: ");

scanf("%d", &size);

// Input array elements

printf("Enter %d elements:\n", size);

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

scanf("%d", &array[i]);

}

// Input the key to search

printf("Enter the key to search: ");

scanf("%d", &key);

// Perform linear search

int result = linearSearch(array, size, key);

// Display the result

if (result != -1) {

printf("Key %d found at index %d.\n", key, result);

} else {

printf("Key %d not found in the array.\n", key);

}

return 0;

}

10) #include <stdio.h>

// Function to swap two numbers using pointers

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

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main() {

int num1, num2;

// Input two numbers

printf("Enter the first number: ");

scanf("%d", &num1);

printf("Enter the second number: ");

scanf("%d", &num2);

// Display the original values

printf("Original values: num1 = %d, num2 = %d\n", num1, num2);

// Call the function to swap values using pointers

swap(&num1, &num2);

// Display the swapped values

printf("Swapped values: num1 = %d, num2 = %d\n", num1, num2);

return 0;

}

11) #include <stdio.h>

int main() {

int arr[] = {1, 2, 3, 4, 5};

int \*ptr = arr;

// a. Increment a pointer

printf("a. Increment a pointer: \*(ptr++) = %d\n", \*(ptr++));

// Reset pointer to the beginning of the array

ptr = arr;

// b. Decrement a pointer

printf("b. Decrement a pointer: \*(ptr--) = %d\n", \*(ptr--));

// Reset pointer to the beginning of the array

ptr = arr;

// c. Add an integer to a pointer

printf("c. Add an integer to a pointer: \*(ptr + 2) = %d\n", \*(ptr + 2));

// d. Subtract an integer from a pointer

printf("d. Subtract an integer from a pointer: \*(ptr - 1) = %d\n", \*(ptr - 1));

// e. Subtract two pointers of the same type

int \*ptr2 = arr + 3;

printf("e. Subtract two pointers of the same type: ptr2 - ptr = %ld\n", ptr2 - ptr);

return 0;

}

12) #include <stdio.h>

// Recursive function to calculate factorial

int factorial(int n) {

if (n == 0 || n == 1) {

return 1;

} else {

return n \* factorial(n - 1);

}

}

int main() {

int num;

// Input the number

printf("Enter a non-negative integer: ");

scanf("%d", &num);

// Check if the number is non-negative

if (num < 0) {

printf("Please enter a non-negative integer.\n");

} else {

// Call the recursive function to calculate factorial

int result = factorial(num);

// Display the result

printf("Factorial of %d = %d\n", num, result);

}

return 0;

}

13) #include <stdio.h>

// Function to swap two numbers using pointers (call by reference)

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

int temp = \*a;

\*a = \*b;

\*b = temp;

}

// Function to increment a number using pointers (call by reference)

void increment(int \*num) {

(\*num)++;

}

int main() {

int num1 = 5, num2 = 10;

printf("Before swapping: num1 = %d, num2 = %d\n", num1, num2);

// Call the swap function to swap values using pointers

swap(&num1, &num2);

printf("After swapping: num1 = %d, num2 = %d\n", num1, num2);

printf("Before incrementing: num1 = %d\n", num1);

// Call the increment function to increment the value using pointers

increment(&num1);

printf("After incrementing: num1 = %d\n", num1);

return 0;

}

14) #include <stdio.h>

#include <stdlib.h>

int main() {

int \*arr;

int size;

// Input the size of the array

printf("Enter the size of the array: ");

scanf("%d", &size);

// Dynamically allocate memory for the array

arr = (int \*)malloc(size \* sizeof(int));

// Check if memory allocation was successful

if (arr == NULL) {

printf("Memory allocation failed. Exiting the program.\n");

return 1;

}

// Input elements for the array

printf("Enter %d elements:\n", size);

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

scanf("%d", &arr[i]);

}

// Display the elements of the array

printf("Elements of the array:\n");

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

printf("%d ", arr[i]);

}

printf("\n");

// Free the dynamically allocated memory

free(arr);

return 0;

}

15) #include <stdio.h>

// Function to calculate the sum of diagonal elements in a square matrix

int sumDiagonal(int matrix[10][10], int size) {

int sum = 0;

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

sum += matrix[i][i]; // Add the diagonal element at position (i, i)

}

return sum;

}

int main() {

int matrix[10][10];

int size;

// Input the size of the square matrix

printf("Enter the size of the square matrix: ");

scanf("%d", &size);

// Input elements for the square matrix

printf("Enter the elements of the square matrix:\n");

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

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

scanf("%d", &matrix[i][j]);

}

}

// Call the function to calculate the sum of diagonal elements

int diagonalSum = sumDiagonal(matrix, size);

// Display the sum of diagonal elements

printf("Sum of diagonal elements: %d\n", diagonalSum);

return 0;

}