**Programs 1 to 10**

**1. Write a C program to perform Matrix Multiplication**

#include <stdio.h>

#define ROW1 3

#define COL1 3

#define ROW2 3

#define COL2 3

void matrixMultiplication(int mat1[ROW1][COL1], int mat2[ROW2][COL2], int result[ROW1][COL2]) {

for (int i = 0; i < ROW1; i++) {

for (int j = 0; j < COL2; j++) {

result[i][j] = 0;

for (int k = 0; k < COL1; k++) {

result[i][j] += mat1[i][k] \* mat2[k][j];

}

}

}

}

void printMatrix(int mat[ROW1][COL2]) {

for (int i = 0; i < ROW1; i++) {

for (int j = 0; j < COL2; j++) {

printf("%d ", mat[i][j]);

}

printf("\n");

}

}

int main() {

int matrix1[ROW1][COL1] = {

{1, 2, 3},

{4, 5, 6},

{7, 8, 9}

};

int matrix2[ROW2][COL2] = {

{9, 8, 7},

{6, 5, 4},

{3, 2, 1}

};

int resultMatrix[ROW1][COL2];

matrixMultiplication(matrix1, matrix2, resultMatrix);

printf("Matrix 1:\n");

printMatrix(matrix1);

printf("Matrix 2:\n");

printMatrix(matrix2);

printf("Result Matrix:\n");

printMatrix(resultMatrix);

return 0;

}

**Output**

Matrix 1:

1 2 3

4 5 6

7 8 9

Matrix 2:

9 8 7

6 5 4

3 2 1

Result Matrix:

30 24 18

84 69 54

138 114 90

**2. Write a C program to find Odd or Even number from a given set of numbers**

#include <stdio.h>

int main() {

int numCount;

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

scanf("%d", &numCount);

int numbers[numCount];

printf("Enter %d numbers:\n", numCount);

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

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

}

printf("\nOdd/Even results:\n");

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

if (numbers[i] % 2 == 0) {

printf("%d is even\n", numbers[i]);

} else {

printf("%d is odd\n", numbers[i]);

}

}

return 0;

}

**Output**

Enter the number of elements: 5

Enter 5 numbers:

10

15

6

7

22

Odd/Even results:

10 is even

15 is odd

6 is even

7 is odd

22 is even

**3. Write a C program to find Factorial of a given number without using Recursion**

#include <stdio.h>

int main() {

int number;

unsigned long long factorial = 1;

printf("Enter a number: ");

scanf("%d", &number);

if (number < 0) {

printf("Factorial is not defined for negative numbers.\n");

} else {

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

factorial \*= i;

}

printf("Factorial of %d is %llu\n", number, factorial);

}

return 0;

}

**Output**

Enter a number: 5

Factorial of 5 is 120

**4. Write a C program to find Fibonacci series without using Recursion**

#include <stdio.h>

int main() {

int n;

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

scanf("%d", &n);

int first = 0, second = 1, next;

printf("Fibonacci Series: %d %d ", first, second);

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

next = first + second;

printf("%d ", next);

first = second;

second = next;

}

    return 0;

}

**Output**

Enter the number of terms: 10

Fibonacci series:

0 1 1 2 3 5 8 13 21 34

**5. Write a C program to find Factorial of a given number using Recursion**

#include <stdio.h>

int factorial(int n) {

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

return 1;

} else {

return n \* factorial(n - 1);

}

}

int main() {

int number;

printf("Enter a number: ");

scanf("%d", &number);

if (number < 0) {

printf("Factorial is not defined for negative numbers.\n");

} else {

int result = factorial(number);

printf("Factorial of %d is %d\n", number, result);

}

return 0;

}

**Output**

Enter a number: 5

Factorial of 5 is 120

**6. Write a C program to find Fibonacci series using Recursion**

#include <stdio.h>

int fibonacci(int n) {

if (n <= 0) {

return 0;

} else if (n == 1) {

return 1;

}

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

}

int main() {

int n;

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

scanf("%d", &n);

printf("Fibonacci Series: ");

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

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

}

printf("\n");

return 0;

}

**Output**

Enter the number of terms: 10

Fibonacci series:

0 1 1 2 3 5 8 13 21 34

**7. Write a C program to implement Array operations such as Insert, Delete and Display**

#include <stdio.h>

#define MAX\_SIZE 100

// Function to display the elements of the array

void display(int arr[], int size) {

printf("Array elements: ");

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

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

}

printf("\n");

}

// Function to insert an element at a specified position

int insert(int arr[], int size, int element, int position) {

if (size >= MAX\_SIZE) {

printf("Array is full. Insertion not possible.\n");

return size;

}

if (position < 0 || position > size) {

printf("Invalid position for insertion.\n");

return size;

}

for (int i = size; i > position; i--) {

arr[i] = arr[i - 1];

}

arr[position] = element;

return size + 1;

}

// Function to delete an element from a specified position

int deleteElement(int arr[], int size, int position) {

if (size <= 0) {

printf("Array is empty. Deletion not possible.\n");

return size;

}

if (position < 0 || position >= size) {

printf("Invalid position for deletion.\n");

return size;

}

for (int i = position; i < size - 1; i++) {

arr[i] = arr[i + 1];

}

return size - 1;

}

int main() {

int arr[MAX\_SIZE];

int size = 0;

int choice, element, position;

do {

printf("\nArray Operations:\n");

printf("1. Insert\n");

printf("2. Delete\n");

printf("3. Display\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter the element to insert: ");

scanf("%d", &element);

printf("Enter the position to insert at: ");

scanf("%d", &position);

size = insert(arr, size, element, position);

break;

case 2:

printf("Enter the position to delete from: ");

scanf("%d", &position);

size = deleteElement(arr, size, position);

break;

case 3:

display(arr, size);

break;

case 4:

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

break;

default:

printf("Invalid choice.\n");

}

} while (choice != 4);

return 0;

}

**Output**

Array Operations:

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice:

**8. Write a C program to search a number using Linear Search method**

#include <stdio.h>

int linearSearch(int arr[], int size, int target) {

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

if (arr[i] == target) {

return i;

}

}

return -1;

}

int main() {

int arr[] = {12, 45, 67, 23, 89, 34, 56, 78, 90};

int size = sizeof(arr) / sizeof(arr[0]);

int target;

printf("Enter the number to search for: ");

scanf("%d", &target);

int result = linearSearch(arr, size, target);

if (result != -1) {

printf("Number found at index %d\n", result);

} else {

printf("Number not found in the array.\n");

}

return 0;

}

**Output**

Enter the number to search for: 90

Number found at index 8

**9. Write a C program to search a number using Binary Search method**

#include <stdio.h>

int binarySearch(int arr[], int size, int target) {

int left = 0;

int right = size - 1;

while (left <= right) {

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

if (arr[mid] == target) {

return mid;

} else if (arr[mid] < target) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return -1; // Return -1

}

int main() {

int arr[] = {12, 23, 34, 45, 56, 67, 78, 89, 90};

int size = sizeof(arr) / sizeof(arr[0]);

int target;

printf("Enter the number to search for: ");

scanf("%d", &target);

int result = binarySearch(arr, size, target);

if (result != -1) {

printf("Number found at index %d\n", result);

} else {

printf("Number not found in the array.\n");

}

return 0;

}

**Output**

Enter the number to search for: 89

Number found at index 7

**10. Write a C program to implement Linked list operations**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

struct Node\* insertNode(struct Node\* head, int data) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = data;

newNode->next = head;

return newNode;

}

struct Node\* deleteNode(struct Node\* head, int data) {

if (head == NULL) {

return NULL;

}

if (head->data == data) {

struct Node\* temp = head;

head = head->next;

free(temp);

return head;

}

struct Node\* curr = head;

while (curr->next != NULL && curr->next->data != data) {

curr = curr->next;

}

if (curr->next != NULL) {

struct Node\* temp = curr->next;

curr->next = temp->next;

free(temp);

}

return head;

}

void displayList(struct Node\* head) {

printf("Linked List: ");

while (head != NULL) {

printf("%d ", head->data);

head = head->next;

}

printf("\n");

}

int main() {

struct Node\* head = NULL;

int choice, data;

do {

printf("\nLinked List Operations:\n");

printf("1. Insert\n");

printf("2. Delete\n");

printf("3. Display\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter the element to insert: ");

scanf("%d", &data);

head = insertNode(head, data);

printf("Element inserted successfully.\n");

break;

case 2:

printf("Enter the element to delete: ");

scanf("%d", &data);

head = deleteNode(head, data);

printf("Element deleted successfully.\n");

break;

case 3:

displayList(head);

break;

case 4:

printf("Exiting program.\n");

break;

default:

printf("Invalid choice. Please enter a valid option.\n");

}

} while (choice != 4);

while (head != NULL) {

struct Node\* temp = head;

head = head->next;

free(temp);

}

return 0;

}

**Output**

Linked List Operations:

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice: