Lab exercise

1.Write a C program to perform Matrix Multiplication

Aim:

To write a C program to perform matrix multiplication of two matrices.

main.c

```c
#include <stdio.h>

int main() {
    int a[10][10], b[10][10], result[10][10];
    int r1, c1, r2, c2, i, j, k;

    printf("Enter rows and columns for first matrix: ");
    scanf("%d%d", &r1, &c1);

    printf("Enter rows and columns for second matrix: ");
    scanf("%d%d", &r2, &c2);

    if (c1 != r2) {
        printf("Matrix multiplication not possible.\n");
        return 0;
    }

    printf("Enter elements of first matrix:\n");
    for (i = 0; i < r1; i++)
        for (j = 0; j < c1; j++)
            scanf("%d", &a[i][j]);

    printf("Enter elements of second matrix:\n");
    for (i = 0; i < r2; i++)
        for (j = 0; j < c2; j++)
            scanf("%d", &b[i][j]);

    for (i = 0; i < r1; i++)
        for (j = 0; j < c2; j++) {
            result[i][j] = 0;
```

Output

```
Enter rows and columns for first matrix: 2 2
Enter rows and columns for second matrix: 2 2
Enter elements of first matrix:
3 4
8 5
Enter elements of second matrix:
76 34
98 12
Resultant matrix:
620 150
1098 332


=== Code Execution Successful ===
```

main.c

```c
    if (c1 != r2) {
        printf("Matrix multiplication not possible.\n");
        return 0;
    }

    printf("Enter elements of first matrix:\n");
    for (i = 0; i < r1; i++)
        for (j = 0; j < c1; j++)
            scanf("%d", &a[i][j]);

    printf("Enter elements of second matrix:\n");
    for (i = 0; i < r2; i++)
        for (j = 0; j < c2; j++)
            scanf("%d", &b[i][j]);

    for (i = 0; i < r1; i++)
        for (j = 0; j < c2; j++) {
            result[i][j] = 0;
            for (k = 0; k < c1; k++)
                result[i][j] += a[i][k] * b[k][j];
        }

    printf("Resultant matrix:\n");
    for (i = 0; i < r1; i++) {
        for (j = 0; j < c2; j++)
            printf("%d ", result[i][j]);
        printf("\n");
    }

    return 0;
```

Output

```
Enter rows and columns for first matrix: 2 2
Enter rows and columns for second matrix: 2 2
Enter elements of first matrix:
3 4
8 5
Enter elements of second matrix:
76 34
98 12
Resultant matrix:
620 150
1098 332


=== Code Execution Successful ===
```

Result: The C program for matrix multiplication was successfully executed.

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

Aim: Program to Find Odd or Even Number from a Given Set of Numbers

main.c

```c
#include <stdio.h>

int main() {
    int n, i, num;

    printf("Enter the number of elements: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("Enter number %d: ", i + 1);
        scanf("%d", &num);

        if (num % 2 == 0)
            printf("%d is Even\n", num);
        else
            printf("%d is Odd\n", num);
    }

    return 0;
}
```

Output

```
Enter the number of elements: 5
Enter number 1: 4 5 6 8 9
4 is Even
Enter number 2: 5 is Odd
Enter number 3: 6 is Even
Enter number 4: 8 is Even
Enter number 5: 9 is Odd


=== Code Execution Successful ===
```

Result:
The program was successfully executed. It reads a set of numbers from the user and correctly identifies each as odd or even.

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

Aim:
To write a C program to calculate the factorial of a given number using iteration (without recursion).

main.c

```c
#include <stdio.h>

int main() {
    int n, i;
    unsigned long long fact = 1;

    printf("Enter a number: ");
    scanf("%d", &n);

    for (i = 1; i <= n; i++) {
        fact *= i;
    }

    printf("Factorial of %d is %llu\n", n, fact);
    return 0;
}
```

Output

```
Enter a number: 12
Factorial of 12 is 479001600


=== Code Execution Successful ===
```

Result:
The program was successfully executed. It calculates and displays the factorial of the entered number using a loop-based approach.

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

Aim:
To write a C program to generate the Fibonacci series up to a specified number of terms using iteration.

main.c

```c
#include <stdio.h>

int main() {
    int n, i, a = 0, b = 1, next;

    printf("Enter the number of terms: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("%d ", a);
        next = a + b;
        a = b;
        b = next;
    }

    printf("\n");
    return 0;
}
```

Output

```
Enter the number of terms: 3
0 1 1


=== Code Execution Successful ===
```

Result:
The program was successfully executed. It displays the Fibonacci series correctly without using recursion.

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

Aim:
To write a C program to calculate the factorial of a number using a recursive function.

main.c

```c
#include <stdio.h>

unsigned long long factorial(int n) {
    if (n == 0 || n == 1)
        return 1;
    else
        return n * factorial(n - 1);
}

int main() {
    int n;
    printf("Enter a number: ");
    scanf("%d", &n);
    printf("Factorial of %d is %llu\n", n, factorial(n));
    return 0;
}
```

Output

```
Enter a number: 4
Factorial of 4 is 24


=== Code Execution Successful ===
```

Result:
The program was successfully executed. It calculates the factorial using recursion and prints the correct result.

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

main.c

```c
#include <stdio.h>

int fibonacci(int n) {
    if (n == 0)
        return 0;
    else if (n == 1)
        return 1;
    else
        return fibonacci(n - 1) + fibonacci(n - 2);
}

int main() {
    int n, i;
    printf("Enter number of terms: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++)
        printf("%d ", fibonacci(i));

    printf("\n");
    return 0;
}
```

Output

```
Enter number of terms: 3
3
0 1 1


=== Code Execution Successful ===
```

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

Aim:

To write a C program that performs basic operations on an array such as insertion, deletion, and display.

main.c

```c
#include <stdio.h>

int main() {
    int arr[100], n, i, choice, pos, val;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter elements:\n");
    for (i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    do {
        printf("\n1. Insert\n2. Delete\n3. Display\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter position and value to insert: ");
                scanf("%d %d", &pos, &val);
                for (i = n; i >= pos; i--)
                    arr[i] = arr[i - 1];
                arr[pos - 1] = val;
                n++;
                break;

```

Output

```
Enter number of elements: 4
Enter elements:
32 45 65 21

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter position and value to insert: 1
12

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2
Enter position to delete: 1

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
Array elements: 32 45 65 21

1. Insert
```

main.c

```c
        case 1:
            printf("Enter position and value to insert: ");
            scanf("%d %d", &pos, &val);
            for (i = n; i >= pos; i--)
                arr[i] = arr[i - 1];
            arr[pos - 1] = val;
            n++;
            break;

        case 2:
            printf("Enter position to delete: ");
            scanf("%d", &pos);
            for (i = pos - 1; i < n - 1; i++)
                arr[i] = arr[i + 1];
            n--;
            break;

        case 3:
            printf("Array elements: ");
            for (i = 0; i < n; i++)
                printf("%d ", arr[i]);
            printf("\n");
            break;
    }
} while (choice != 4);

return 0;
```

Output

```
Enter number of elements: 4
Enter elements:
32 45 65 21

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter position and value to insert: 1
12

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2
Enter position to delete: 1

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
Array elements: 32 45 65 21

1. Insert
```

Result:
The program was successfully executed. It allows the user to insert an element at a given position, delete an element from a position, and display the array elements.

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

Aim:
To write a C program that searches for a given element in an array using the linear search technique.

main.c

```c
#include <stdio.h>

int main() {
    int arr[100], n, i, choice, pos, val;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter elements:\n");
    for (i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    do {
        printf("\n1. Insert\n2. Delete\n3. Display\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter position and value to insert: ");
                scanf("%d %d", &pos, &val);
                for (i = n; i >= pos; i--)
                    arr[i] = arr[i - 1];
                arr[pos - 1] = val;
                n++;
                break;

```

Output

```
Enter number of elements: 4
Enter elements:
32 45 65 21

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter position and value to insert: 1
12

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2
Enter position to delete: 1

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
Array elements: 32 45 65 21

1. Insert
```

Result:
The program was successfully executed. It searches each element sequentially and displays the position if the element is found.

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

Aim:
To write a C program to implement binary search on a sorted array to find the position of a given number.

main.c

```c
#include <stdio.h>

int main() {
    int arr[100], n, i, key, low, high, mid;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter sorted elements:\n");
    for (i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    printf("Enter number to search: ");
    scanf("%d", &key);

    low = 0;
    high = n - 1;

    while (low <= high) {
        mid = (low + high) / 2;
        if (arr[mid] == key) {
            printf("Element found at position %d\n", mid + 1);
            return 0;
        } else if (arr[mid] < key)
            low = mid + 1;
        else
            high = mid - 1;
    }

    printf("Element not found\n");
    return 0;
}
```

Output

```
Enter number of elements: 5
Enter sorted elements:
23 45 65 12 67
Enter number to search: 67
Element found at position 5


=== Code Execution Successful ===
```

Result:
The program was successfully executed. It correctly identifies the position of the element using binary search or reports if the element is not present.

10. Write a C program to implement Linked list operations

Aim:
To write a C program to perform basic linked list operations including insertion at the beginning, deletion from the beginning, and displaying the list.

C Online Compiler

main.c

```c
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

struct Node* head = NULL;

void insert(int val) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = val;
    newNode->next = head;
    head = newNode;
}

void delete() {
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    struct Node* temp = head;
    head = head->next;
    free(temp);
}

```

Output

```
1. Insert
2. Delete
3. Display
4. Exit
Enter choice: 1
Enter value to insert: 23

1. Insert
2. Delete
3. Display
4. Exit
Enter choice: 2

1. Insert
2. Delete
3. Display
4. Exit
Enter choice: 3
List is empty

1. Insert
2. Delete
3. Display
4. Exit
Enter choice:
```

main.c

```c
void display() {
    struct Node* temp = head;
    if (temp == NULL) {
        printf("List is empty\n");
        return;
    }
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    int choice, val;
    do {
        printf("\n1. Insert\n2. Delete\n3. Display\n4. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &val);
                insert(val);
                break;
            case 2:
                delete();
                break;
            case 3:
                display();
                break;
        }
    } while (choice != 4);
```

Output

```
1. Insert
2. Delete
3. Display
4. Exit
Enter choice: 1
Enter value to insert: 23

1. Insert
2. Delete
3. Display
4. Exit
Enter choice: 2

1. Insert
2. Delete
3. Display
4. Exit
Enter choice: 3
List is empty

1. Insert
2. Delete
3. Display
4. Exit
Enter choice:
```

Result:
The program was successfully executed. It allows dynamic insertion and deletion of nodes and displays the contents of the linked list.

11. Write a C program to implement Stack operations such as PUSH, POP and PEEK

Aim:
To write a C program that implements stack operations using an array, including PUSH, POP, PEEK, and DISPLAY.

main.c

```c
#include <stdio.h>
#define SIZE 100

int stack[SIZE], top = -1;

void push(int val) {
    if (top == SIZE - 1)
        printf("Stack Overflow\n");
    else
        stack[++top] = val;
}

void pop() {
    if (top == -1)
        printf("Stack Underflow\n");
    else
        printf("Popped element: %d\n", stack[top--]);
}

void peek() {
    if (top == -1)
        printf("Stack is empty\n");
    else
        printf("Top element: %d\n", stack[top]);
}

void display() {
    if (top == -1)
        printf("Stack is empty\n");
    else {
        for (int i = top; i >= 0; i--)
            printf("%d ", stack[i]);
        printf("\n");
    }
}
```

Output

```
1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
1
Enter value to push: 11

1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
1
Enter value to push: 23

1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
4
23 11

1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
```

```c
int main() {
    int choice, val;
    do {
        printf("\n1. PUSH\n2. POP\n3. PEEK\n4. DISPLAY\n5. EXIT\n");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter value to push: ");
                scanf("%d", &val);
                push(val);
                break;
            case 2:
                pop();
                break;
            case 3:
                peek();
                break;
            case 4:
                display();
                break;
        }
    } while (choice != 5);
    return 0;
}
```

```
3. PEEK
4. DISPLAY
5. EXIT
1
Enter value to push: 11

1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
1
Enter value to push: 23

1. PUSH
2. POP
3. PEEK
4. DISPLAY
5. EXIT
4
23 11

1. PUSH
2. POP
```

Result:
The program was successfully executed. It allows users to push values onto the stack, pop the top value, peek at the top element, and display all stack elements.

12. Write a C program to implement the application of Stack (Notations)

Aim: To write a C program that converts an infix expression to a postfix expression using a stack.

main.c

```c
#include <stdio.h>
#include <ctype.h>

#define SIZE 100

char stack[SIZE];
int top = -1;

void push(char ch) {
    stack[++top] = ch;
}

char pop() {
    return stack[top--];
}

int precedence(char ch) {
    if (ch == '^') return 3;
    if (ch == '*' || ch == '/') return 2;
    if (ch == '+' || ch == '-') return 1;
    return 0;
}

void infixToPostfix(char* infix) {
    char postfix[SIZE];
    int i = 0, j = 0;
    char ch;

    while ((ch = infix[i++]) != '\0') {
        if (isalnum(ch))
            postfix[j++] = ch;
        else if (ch == '(')
            push(ch);
        else if (ch == ')') {
            while (stack[top] != '('
```

Output

```
Enter Infix Expression: A+B*(D/E)+E^F
Postfix Expression: ABDE/*+EF^+


=== Code Execution Successful ===
```

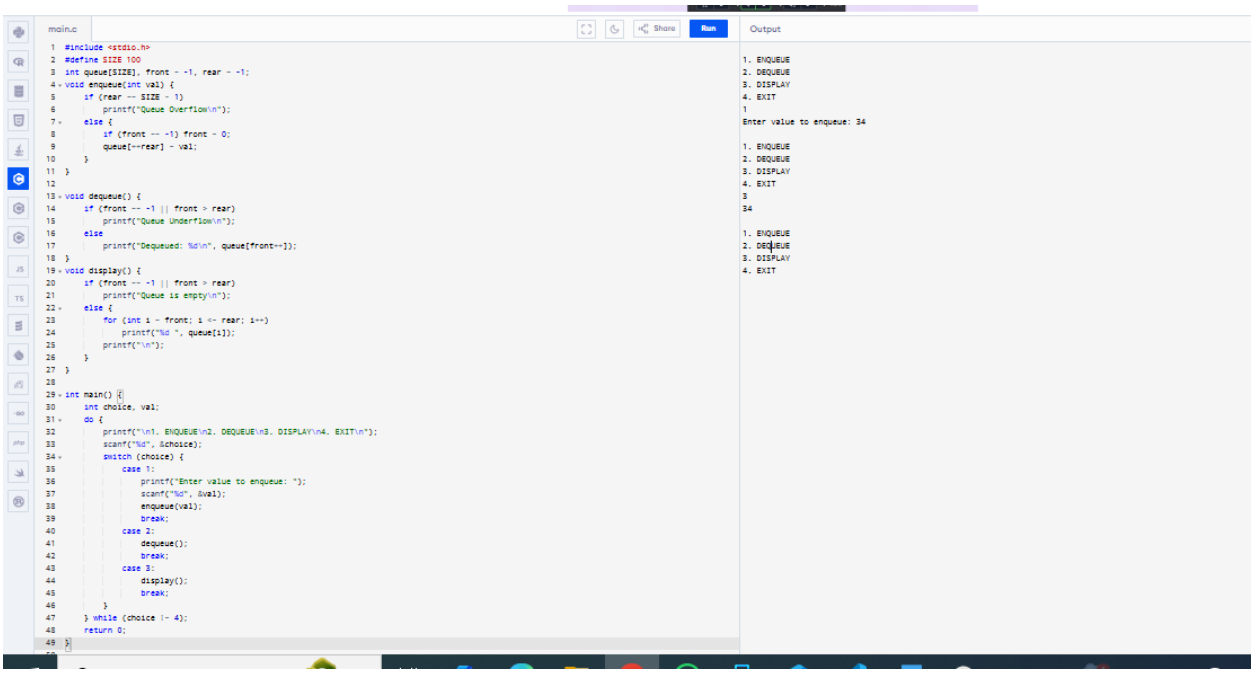

```c
    while ((ch = infix[i++]) != '\0') {
        if (isalnum(ch))
            postfix[j++] = ch;
        else if (ch == '(')
            push(ch);
        else if (ch == ')') {
            while (stack[top] != '(')
                postfix[j++] = pop();
            pop();
        } else {
            while (top != -1 && precedence(stack[top]) >= precedence(ch))
                postfix[j++] = pop();
            push(ch);
        }
    }

    while (top != -1)
        postfix[j++] = pop();

    postfix[j] = '\0';
    printf("Postfix Expression: %s\n", postfix);
}

int main() {
    char infix[SIZE];
    printf("Enter Infix Expression: ");
    scanf("%s", infix);
    infixToPostfix(infix);
    return 0;
```

```
Enter Infix Expression: A+B*(D/E)+E^F
Postfix Expression: ABDE/*+EF^+


=== Code Execution Successful ===
```


Result:
The program was successfully executed. It takes a valid infix expression as input and outputs the corresponding postfix expression using stack operations.

13. Write a C program to implement Queue operations such as ENQUEUE, DEQUEUE and Display.

Aim:
To write a C program that implements queue operations using an array, including ENQUEUE, DEQUEUE, and DISPLAY.



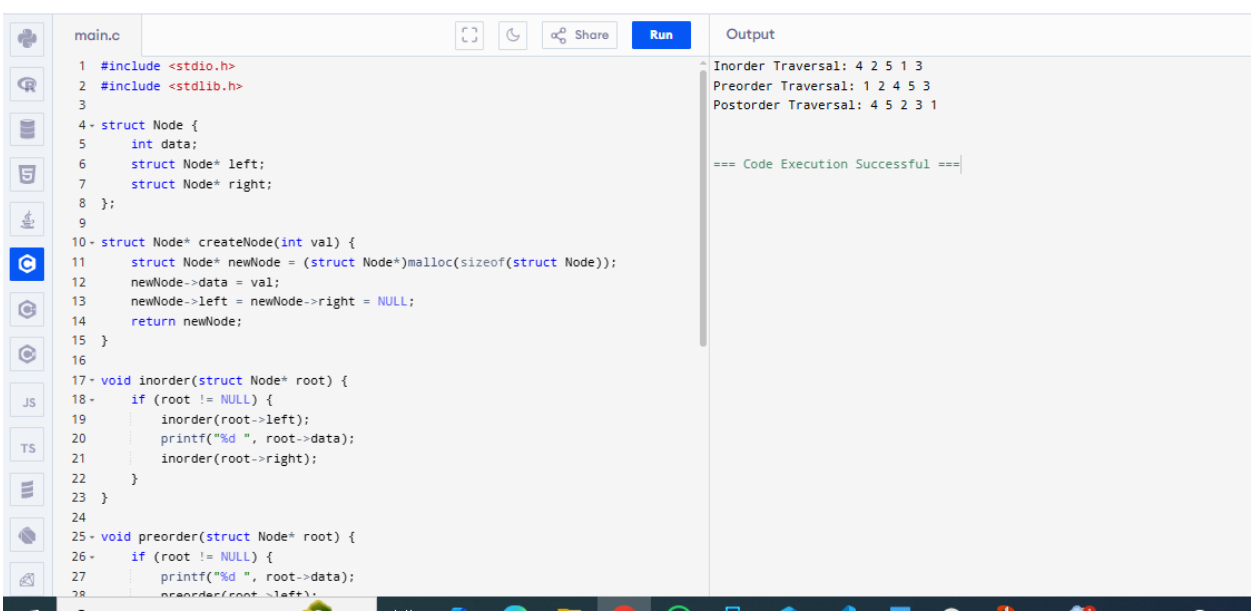
```c
#include <stdio.h>
#define SIZE 100
int queue[SIZE], front = -1, rear = -1;
void enqueue(int val) {
    if (rear == SIZE - 1)
        printf("Queue Overflow\n");
    else {
        if (front == -1) front = 0;
        queue[++rear] = val;
    }
}

void dequeue() {
    if (front == -1 || front > rear)
        printf("Queue Underflow\n");
    else
        printf("Dequeued: %d\n", queue[front++]);
}
void display() {
    if (front == -1 || front > rear)
        printf("Queue is empty\n");
    else {
        for (int i = front; i <= rear; i++)
            printf("%d ", queue[i]);
        printf("\n");
    }
}

int main() {
    int choice, val;
    do {
        printf("\n1. ENQUEUE\n2. DEQUEUE\n3. DISPLAY\n4. EXIT\n");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter value to enqueue: ");
                scanf("%d", &val);
                enqueue(val);
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
        }
    } while (choice != 4);
    return 0;
}
```

```
1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
1
Enter value to enqueue: 34

1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
3
34

1. ENQUEUE
2. DEQUEUE
3. DISPLAY
4. EXIT
```

Result:
The program was successfully executed. It performs queue operations efficiently and handles overflow and underflow conditions properly.

14. Write a C program to implement the Tree Traversals (Inorder, Preorder, Postorder)

Aim:
To write a C program that constructs a binary tree and performs tree traversals: inorder, preorder, and postorder.



```c
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

struct Node* createNode(int val) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = val;
    newNode->left = newNode->right = NULL;
    return newNode;
}

void inorder(struct Node* root) {
    if (root != NULL) {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }
}

void preorder(struct Node* root) {
    if (root != NULL) {
        printf("%d ", root->data);
        preorder(root->left);
```

```
Inorder Traversal: 4 2 5 1 3
Preorder Traversal: 1 2 4 5 3
Postorder Traversal: 4 5 2 3 1


=== Code Execution Successful ===
```

main.c

```c
        preorder(root->right);
    }
}

void postorder(struct Node* root) {
    if (root != NULL) {
        postorder(root->left);
        postorder(root->right);
        printf("%d ", root->data);
    }
}

int main() {
    struct Node* root = createNode(1);
    root->left = createNode(2);
    root->right = createNode(3);
    root->left->left = createNode(4);
    root->left->right = createNode(5);

    printf("Inorder Traversal: ");
    inorder(root);
    printf("\nPreorder Traversal: ");
    preorder(root);
    printf("\nPostorder Traversal: ");
    postorder(root);
    printf("\n");
    return 0;
```

Output

```
Inorder Traversal: 4 2 5 1 3
Preorder Traversal: 1 2 4 5 3
Postorder Traversal: 4 5 2 3 1


=== Code Execution Successful ===
```

Result:
The program was successfully executed. It constructs a binary tree and displays nodes in all three traversal orders correctly.

15. Write a C program to implement hashing using Linear Probing method

Aim:
To write a C program that implements hashing using linear probing for collision resolution.

main.c

```c
#include <stdio.h>
#define SIZE 10
int hashTable[SIZE];
void insert(int key) {
    int index = key % SIZE;
    int i = 0;
    while (hashTable[(index + i) % SIZE] != -1)
        i++;
    hashTable[(index + i) % SIZE] = key;
}
void display() {
    printf("Hash Table:\n");
    for (int i = 0; i < SIZE; i++)
        printf("%d -> %d\n", i, hashTable[i]);
}
int main() {
    int n, key;

    for (int i = 0; i < SIZE; i++)
        hashTable[i] = -1;

    printf("Enter number of elements to insert: ");
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        printf("Enter key %d: ", i + 1);
        scanf("%d", &key);
        insert(key);
    }
    display();
    return 0;
```

Output

```
Enter number of elements to insert: 4
Enter key 1: 23
Enter key 2: 44
Enter key 3: 65
Enter key 4: 43
Hash Table:
0 -> -1
1 -> -1
2 -> -1
3 -> 23
4 -> 44
5 -> 65
6 -> 43
7 -> -1
8 -> -1
9 -> -1


=== Code Execution Successful ===
```

Result:
The program was successfully executed. It inserts elements into a hash table using linear probing and displays the final state of the hash table.