Assignment-04
Name : Parth Sali
Roll No : 23167
Batch : H9

Code :

```
#include <bits/stdc++.h>
using namespace std;

class Node
{
public:
    char data;
    Node *left;
    Node *right;

    Node()
    {
        this->data = 0;
        left = NULL;
        right = NULL;
    }
    Node(char data)
    {
        this->data = data;
        this->left = NULL;
        this->right = NULL;
    }
};

void inOrderWithReccursion(Node *root)
{
    if (root == NULL)
    {
        return;
    }
    else
    {
        inOrderWithReccursion(root->left);
        cout << root->data << " ";
        inOrderWithReccursion(root->right);
    }
}

void inOrderWithoutReccursion(Node *root)
{
    stack<Node *> st;
    Node *current = root;
    while (current != NULL || !st.empty())
    {
        while (current != NULL)
        {
            st.push(current);
            current = current->left;
        }
```

```
        current = st.top();

        st.pop();

        cout << current->data << " ";
        current = current->right;
    }
    cout << endl;
}

void preOrderWithReccursion(Node *root)
{
    if (root == NULL)
    {
        return;
    }
    else
    {
        cout << root->data << " ";
        preOrderWithReccursion(root->left);
        preOrderWithReccursion(root->right);
    }
}

void preOrderWithoutReccursion(Node *root)
{
    stack<Node *> st;

    if (root == NULL)
    {
        return;
    }
    st.push(root);
    while (!st.empty())
    {
        Node *temp = st.top();
        cout << temp->data << " ";
        st.pop();

        if (temp->right != NULL)
        {

            st.push(temp->right);
        }
        if (temp->left != NULL)
        {
            st.push(temp->left);
        }
    }
    cout << endl;
}

void postOrderWithReccursion(Node *root)
{
```

```
    if (root == NULL)
    {
        return;
    }
    else
    {
        postOrderWithReccursion(root->left);
        postOrderWithReccursion(root->right);
        cout << root->data << " ";
    }
}

void postOrderWithoutReccursion(Node *root)
{

    if (root == NULL)
    {
        return;
    }
    stack<Node *> s1, s2;
    Node *temp = root;
    s1.push(temp);
    while (!s1.empty())
    {
        temp = s1.top();
        s1.pop();

        s2.push(temp);

        if (temp->left != NULL)
        {
            s1.push(temp->left);
        }
        if (temp->right != NULL)
        {
            s1.push(temp->right);
        }
    }

    while (!s2.empty())
    {
        cout << s2.top()->data << " ";
        s2.pop();
    }
    cout << endl;
}

int main()
{
    cout << endl;
    cout << "Enter Postfix Expression :  ";
    string s;
    cin >> s;
    stack<Node *> tree;
    cout << endl;
```

```
for (int i = 0; i < s.size(); i++)
{
   if (s[i] >= 48)
   {
      Node *temp = new Node(s[i]);
      tree.push(temp);
   }
   else
   {
      Node *operation = new Node(s[i]);
      Node *second = tree.top();
      tree.pop();
      Node *first = tree.top();
      tree.pop();
      operation->left = first;
      operation->right = second;
      tree.push(operation);
   }
}

while (true)
{

   cout << "1. Inorder Traversal with recursion." << endl;
   cout << "2. Preorder Traversal with recursion." << endl;
   cout << "3. Postorder Traversal with recursion." << endl;
   cout << "4. Inorder Traversal without recursion." << endl;
   cout << "5. Preorder Traversal without recursion." << endl;
   cout << "6. Postorder Traversal without recursion." << endl;
   cout << "7. Exit" << endl;
   cout << endl;
   int choice;
   cout << "Enter Your Choice : ";
   cin >> choice;
   cout << endl;

   switch (choice)
   {
   case 1:
      cout << "Inorder : ";
      inOrderWithReccursion(tree.top());
      cout << endl;
      cout << endl;
      break;
   case 2:
      cout << "Preorder : ";
      preOrderWithReccursion(tree.top());
      cout << endl;
      cout << endl;
      break;
   case 3:
      cout << "Postorder : ";
      postOrderWithReccursion(tree.top());
      cout << endl;
      cout << endl;
```

```
            break;
        case 4:
            cout << "Inorder : ";
            inOrderWithoutReccursion(tree.top());
            cout << endl;
            break;
        case 5:
            cout << "Preorder : ";
            preOrderWithoutReccursion(tree.top());
            cout << endl;
            break;
        case 6:
            cout << "Postorder : ";
            postOrderWithoutReccursion(tree.top());
            cout << endl;
            break;
        case 7:
            cout << "Exiting the program..." << endl;
            exit(0);
        default:
            cout << "Enter Valid Choice..." << endl;
            break;
        }
    }
}
```

OUTPUT :

```
Enter Postfix Expression :  23+45+*

1. Inorder Traversal with recursion.
2. Preorder Traversal with recursion.
3. Postorder Traversal with recursion.
4. Inorder Traversal without recursion.
5. Preorder Traversal without recursion.
6. Postorder Traversal without recursion.
7. Exit

Enter Your Choice : 1

Inorder : 2 + 3 * 4 + 5

1. Inorder Traversal with recursion.
2. Preorder Traversal with recursion.
3. Postorder Traversal with recursion.
4. Inorder Traversal without recursion.
5. Preorder Traversal without recursion.
6. Postorder Traversal without recursion.
7. Exit

Enter Your Choice : 2

Preorder : * + 2 3 + 4 5

1. Inorder Traversal with recursion.
```

2. Preorder Traversal with recursion.
3. Postorder Traversal with recursion.
4. Inorder Traversal without recursion.
5. Preorder Traversal without recursion.
6. Postorder Traversal without recursion.
7. Exit

Enter Your Choice : 3

Postorder : 2 3 + 4 5 + *

1. Inorder Traversal with recursion.
2. Preorder Traversal with recursion.
3. Postorder Traversal with recursion.
4. Inorder Traversal without recursion.
5. Preorder Traversal without recursion.
6. Postorder Traversal without recursion.
7. Exit

Enter Your Choice : 4

Inorder : 2 + 3 * 4 + 5

1. Inorder Traversal with recursion.
2. Preorder Traversal with recursion.
3. Postorder Traversal with recursion.
4. Inorder Traversal without recursion.
5. Preorder Traversal without recursion.
6. Postorder Traversal without recursion.
7. Exit

Enter Your Choice : 5

Preorder : * + 2 3 + 4 5

1. Inorder Traversal with recursion.
2. Preorder Traversal with recursion.
3. Postorder Traversal with recursion.
4. Inorder Traversal without recursion.
5. Preorder Traversal without recursion.
6. Postorder Traversal without recursion.
7. Exit

Enter Your Choice : 6

Postorder : 2 3 + 4 5 + *

1. Inorder Traversal with recursion.
2. Preorder Traversal with recursion.
3. Postorder Traversal with recursion.
4. Inorder Traversal without recursion.
5. Preorder Traversal without recursion.
6. Postorder Traversal without recursion.
7. Exit

Enter Your Choice : 7

Exiting the program..