```
/*
Subject : DSA Laboratory
Practical No: 05
Title : A C++ Program to Create and Display an Expression Tree
        Input : Postfix Expression
        Output: Preorder, Inorder and Postorder Traversal of Expression Tree
*/

            //.........Header Files
#include <iostream>
using namespace std;

            //.........Input to Program
char postfix[10];


            //.........Node of Expression Tree
struct Node
{
   char data;
   struct Node *left;
   struct Node *right;
}*Root;

            //.......Stack to store Pointers of Nodes
struct Node* stack[5];
int top = -1;

            //......To push Pointers in Stack
void push_stk(struct Node *newnode)
{
   top++;
   stack[top] = newnode;
}


            //......To pop Pointers from Stack
struct Node* pop_stk()
{
   struct Node *temp;

   temp = stack[top];
   top--;

   return temp;
}
```

```
          //.........Function to Create New Node
struct Node* create_Node(char val)
{
   struct Node *Newnode;

   Newnode = new struct Node;

   Newnode->data = val;
   Newnode->left = NULL;
   Newnode->right = NULL;

   return Newnode;
}

          //.........Function to Create an Expression Tree
void create_Exptree()
{
   int i;

   struct Node *Newnode;

   cout<<"\n\n Enter the Postfix Expression: ";
   cin>>postfix;

   for(i=0; postfix[i] != '\0'; i++)
   {                                                    //.....If Operand

      if(postfix[i] == 'a' || postfix[i] == 'b' || postfix[i] == 'c' || postfix[i] == 'd')
      {
                    //........Create New Node for Operand
         Newnode = create_Node(postfix[i]);
                    //........Push Operand in Stack
         push_stk(Newnode);
      }
                                                        //.....If Operator
      if(postfix[i] == '+' || postfix[i] == '-' || postfix[i] == '*' || postfix[i] == '/')
      {
                    //........Create New Node for Operator
         Newnode = create_Node(postfix[i]);
                    //........Pop An Operand from stack and attach as Right Child
         Newnode->right = pop_stk();
                    //........Pop An Operand from stack and attach as left Child
         Newnode->left = pop_stk();

         push_stk(Newnode);
      }
   }
   if(Root == NULL)
   {               //........Pop a Pointer from Stack and Assign to Root.
      Root = pop_stk();
      cout<<"\n\t Expression Tree is Ready Now...!!!";
   }
}

         //.........Function to display Expression Tree in Preorder
void preorder_ExpTree(struct Node *root)
{
   if(root)
   {
      cout<<" "<<root->data;          //....Data
      preorder_ExpTree(root->left);   //....Left
      preorder_ExpTree(root->right);  //....Right
   }
}

         //.........Function to display Expression Tree in Inorder
void inorder_ExpTree(struct Node *root)
{
   if(root)
   {
      inorder_ExpTree(root->left);    //....Left
      cout<<" "<<root->data;          //....Data
      inorder_ExpTree(root->right);   //....Right
   }
}

         //.........Function to display Expression Tree in Postorder
void postorder_ExpTree(struct Node *root)
{
   if(root)
   {
      postorder_ExpTree(root->left);  //....Left
      postorder_ExpTree(root->right); //....Right
      cout<<" "<<root->data;          //....Data
   }
}

         //.........Main Function
int main()
{
   cout<<"---------*** A C++ Program to Create and Display an Expression Tree***
---------";

   Root = NULL;
   create_Exptree();

   cout<<"\n\n Preorder Traversal of Expression Tree: ";
   preorder_ExpTree(Root);

   cout<<"\n\n Inorder Traversal of Expression Tree: ";
   inorder_ExpTree(Root);

   cout<<"\n\n Postorder Traversal of Expression Tree: ";
   postorder_ExpTree(Root);

   return 0;
}
```

```
/*-------------------------OUTPUT----------------------------


---------*** A C++ Program to Create and Display an Expression Tree*** ---------

 Enter the Postfix Expression: ab+

   Expression Tree is Ready Now...!!!

 Preorder Traversal of Expression Tree:  + a b

 Inorder Traversal of Expression Tree:  a + b

 Postorder Traversal of Expression Tree:  a b +




 ---------*** A C++ Program to Create and Display an Expression Tree*** ---------

 Enter the Postfix Expression: ab+cd-*

   Expression Tree is Ready Now...!!!

 Preorder Traversal of Expression Tree:  * + a b - c d

 Inorder Traversal of Expression Tree:  a + b * c - d

 Postorder Traversal of Expression Tree:  a b + c d - *
```

```
---------*** A C++ Program to Create and Display an Expression Tree*** ---------

 Enter the Postfix Expression: ab+c*

        Expression Tree is Ready Now...!!!

 Preorder Traversal of Expression Tree:  * + a b c

 Inorder Traversal of Expression Tree:  a + b * c

 Postorder Traversal of Expression Tree:  a b + c *


...Program finished with exit code 0
Press ENTER to exit console.

*/
```