```
//BINARY SEARCH TREE

#include<iostream>
using namespace std;
class Node
{
public:
char data;
Node *left,*right;
};
class Tree
{
public:
Node *root;
Tree()
{
root=NULL;
}
void inorder_recursive(Node *);
void preorder_recursive(Node *);
void postorder_recursive(Node *);
Node *expression();
};
Node * Tree::expression()
{
Node *temp;
int i,top=-1;
char exp[20];
Node *stack[10];
int flag[10];
   cout<<"Enter the expression:";
   cin>>exp;
for(i=0;exp[i]!='\0';i++)
{
if(exp[i]>='a' && exp[i]<='z')
{
temp=new Node;
temp->data=exp[i];
temp->left=temp->right=NULL;
stack[++top]=temp;
}
else
{
root->data=exp[i];
root->left=root->right=NULL;
root->right=stack[top--];
root->left=stack[top--];
stack[++top]=root;
}
}
root=stack[top--];
return root;
}
void Tree:: inorder_recursive(Node *root)
{
if(root!=NULL)
{
inorder_recursive(root->left);
cout<<root->data;
inorder_recursive(root->right);
}
}
void Tree:: preorder_recursive(Node *root)
{
if(root!=NULL)
{
cout<<root->data;
preorder_recursive(root->left);
preorder_recursive(root->right);
}
}
void Tree:: postorder_recursive(Node *root)
{
if(root!=NULL)
{
postorder_recursive(root->left);
postorder_recursive(root->right);
cout<<root->data;
}
}
int main()
{
```

```
Tree s;
s.root=s.expression();
int ch,ans;

cout<<"\n1.Recursive function for inorder";
cout<<"\n2.Recursive function for preorder";
cout<<"\n3.Recursive function for postorder";

do
{
cout<<"\nENTER YOUR CHOICE:";
cin>>ch;
switch(ch)
{
case 1:
cout<<"\n Inorder Exp with Recursive =>";
s.inorder_recursive(s.root);
break;
case 2:
cout<<"\n preorder Exp with Recursive =>";
s.preorder_recursive(s.root);
break;
case 3:
cout<<"\n postorder Exp with Recursive =>";
s.postorder_recursive(s.root);
break;
}
cout<<"\n do you want to continue";
cin>>ans;
}
while(ans==1);
return 0;
}

 /*OUTPUT

Enter the expression:abc*+e*

1.Recursive function for inorder
2.Recursive function for preorder
3.Recursive function for postorder
```

```
ENTER YOUR CHOICE:1
Inorder Exp with Recursive =>a+b*c*e
do you want to continue1
ENTER YOUR CHOICE:2
preorder Exp with Recursive =>*+a*bce
do you want to continue1
ENTER YOUR CHOICE:3
postorder Exp with Recursive =>abc*+e*
do you want to continue0 */
```