```
#include <iostream>
using namespace std;
struct node
{
char data;
node *left,*right;
};
class stack
{
node *data[30];
int top;
public:
stack()
{
top=-1;
}
int empty()
{
if(top==-1)
return 1;
return 0;
}
void push(node*p)
{
data[++top]=p;
}
node *pop()
{
return(data[top--]);
}
};
class expression
{
public:
node*create_from_post(char []);
void inorder(node *t);
void preorder(node *t);
void postorder(node *t);
void non_rec_inorder(node *t1);
void non_rec_preorder(node *t1);
void non_rec_postorder(node *t1);
};
node*expression::create_from_post(char exp[])
{
char c;
stack s;
node *top,*t1,*t2,*root;
int i;
for(i=0;exp[i]!='\0';i++)
{
c=exp[i];
if(isalnum(c))
{
top=new node;
top->left=NULL;
top->right=NULL;
top->data=c;
s.push(top);
}
else
```

```
{
t2=s.pop();

t1=s.pop();
top=new node;
top->data=c;
top->left=t1;
top->right=t2;
s.push(top);
}
}
root=s.pop();
return (root);
}
void expression::inorder(node *t)
{
if(t!=NULL)
{
inorder(t->left);
cout<<" "<<t->data;
inorder(t->right);
}
}
void expression::preorder(node *t)
{
if(t!=NULL)
{
cout<<" "<<t->data;
preorder(t->left);
preorder(t->right);
}
}
void expression::postorder(node *t)
{
if(t!=NULL)
{
postorder(t->left);
postorder(t->right);
cout<<" "<<t->data;
}
}
void expression::non_rec_inorder(node *t1)
{
stack s;
while(!s.empty()||t1!=NULL)
{
while(t1!=NULL)
{
s.push(t1);
t1=t1->left;
}
t1=s.pop();
cout<<" "<<t1->data;
t1=t1->right;
}
}
void expression::non_rec_preorder(node *t1)
{
stack s;
while(!s.empty()||t1!=NULL)
```

```
{
while(t1!=NULL)
{
cout<<" "<<t1->data;
s.push(t1);
t1=t1->left;

}
t1=s.pop();
t1=t1->right;
}
}
void expression::non_rec_postorder(node *t1)
{
stack s;
int i=0;
char str[30];
while(!s.empty()||t1!=NULL)
{
str[i++]=t1->data;
s.push(t1);
t1=t1->right;
}
t1=s.pop();
t1=t1->left;
while(--i>=0)
{
cout<<" "<<str[i];
}
}
int main()
{
int ch;
char postfix[30];
node*root;
root=NULL;
expression e;
do
{
cout<<"\n1.Create Expression tree(from postfix)\n2.Recursive";

cout<<"Traversal\n3.Non recursive Traversal";

cin>>ch;
switch(ch)
{
case 1:
cout<<"\nEnter postfix expression";
cin>>postfix;
//root=e.create_from_post(postfix);
root=e.create_from_post(postfix);
break;
case 2:
cout<<"\nPreorder Traversal";
e.preorder(root);
cout<<"\nInorder Traversal";
e.inorder(root);
cout<<"\nPostorder Traversal";
e.postorder(root);
break;
```

```
case 3:
cout<<"\n inorder Traversal";
e.non_rec_inorder(root);
cout<<"\n preorder Traversal";
e.non_rec_preorder(root);
cout<<"\nPostorder Traversal";
e.non_rec_postorder(root);
break;
}
cout<<"\nDo you want to continue?(1/0)";
cin>>ch;
}while(ch==1);
}
```

OUTPUT

```
1.Create Expression tree(from postfix)
2.RecursiveTraversal
3.Non recursive Traversal1

Enter postfix expression23+56-

Do you want to continue?(1/0)1

1.Create Expression tree(from postfix)
2.RecursiveTraversal
3.Non recursive Traversal2

Preorder Traversal - 5 6
Inorder Traversal 5 - 6
Postorder Traversal 5 6 -
Do you want to continue?(1/0)1

1.Create Expression tree(from postfix)
2.RecursiveTraversal
3.Non recursive Traversal3

 inorder Traversal 5 - 6
 preorder Traversal - 5 6
```