//Write C++ program to implement tree traversal (Inorder, Preorder , Postorder).

```
#include <iostream>

using namespace std;

struct ver {

int data;

ver *left, *right;

};

class tree {

public:

ver* create(int, ver*);

void in(ver*);

void post(ver*);

void pre(ver*);

};

ver* tree::create(int c, ver* node) {

if (node == NULL) {

node = new ver;

node->data = c;

node->left = NULL;

node->right = NULL;

return node;

}

 else {

if (c < node->data)

node->left = create(c, node->left);
```

```
else
node->right = create(c, node->right);
return node;
}
}
void tree::in(ver* node) {
if (node) {
in(node->left);
cout << node->data << "\t";
in(node->right);
}
}
void tree::pre(ver* node) {
if (node) {
cout << node->data << "\t";
pre(node->left);
pre(node->right);
}
 }
void tree::post(ver* node) {
if (node) {
post(node->left);
post(node->right);
cout << node->data << "\t";
}
```

```
}
int main() {
tree t;
ver* r = NULL;
int n, ch;
cout << "\n 1: Insert 2: Inorder 3: Preorder 4: Postorder 5: Exit :\n";
while (ch != 5) {
cout << "\nEnter Choice:";
cin >> ch;
switch (ch) {
case 1:
cout << "\nEnter Node:";
cin >> n;
 r = t.create(n, r);
break;
case 2:
cout << "\nInorder Traversal:";
t.in(r);
break;
case 3:
cout << "\nPreorder Traversal:";
t.pre(r);
break;
case 4:
cout << "\nPostorder Traversal:";
```

```
t.post(r);
break;
case 5:
return 0;
}
}
}
```

//OUTPUT

```
1: Insert 2: Inorder 3: Preorder 4: Postorder 5: Exit :
Enter Choice:1
Enter Node:10
Enter Choice:1
Enter Node:5
Enter Choice:1
Enter Node:15
Enter Choice:2
Inorder Traversal:5     10      15
Enter Choice:3
Preorder Traversal:10   5       15
Enter Choice:4
Postorder Traversal:5   15      10
Enter Choice:5
EXIT
```