```
#include <iostream>
using namespace std;
struct node {
int data;
node* l;
node* r;
node(int value) {
data = value;
l = nullptr;
r = nullptr;
}
};
node* create(int value) {
return new node(value);
}
node* insertNode(node* root, int value) {
if (root == nullptr) {
return create(value);
} else {
if (root->data > value) {
root->l = insertNode(root->l, value);
} else if (value > root->data) {
root->r = insertNode(root->r, value);
}
}
return root;
}
node* findMin(node* root) {
while (root->l != nullptr) {
root = root->l;
}
cout<<root->data;
return root;}
node* search(node* root, int target) {
if (root == nullptr || root->data == target) {
return root;
}
if (target < root->data) {
return search(root->l, target);
}
else {
return search(root->r, target);
}
}
int lp(node* root) {
if (root == nullptr) {
return 0;
}
else {
int leftp = lp(root->l);
int rightp = lp(root->r);
return max(leftp, rightp) + 1;
```

```
}
}
void swap(node*root){
if(root==NULL){
return ;
}
node*temp=root->l;
root->l=root->r;
root->r=temp;
swap(root->l);
swap(root->r);
}
void inorder(node* root) {
if (root != nullptr) {
inorder(root->l);
cout << root->data << " ";
inorder(root->r);
}
}
int main(){
node*root=NULL;
int target;
int choice;do{
cout<<endl;
cout<<"performing BST operation"<<endl;
cout<<"1.for insert element in tree"<<endl;
cout<<"2. for searching elemnent in tree"<<endl;
cout<<"3 for finding minimum element in tree"<<endl;
cout<<"4 for finding longest path"<<endl;
cout<<"5 for swapping the elements of tree"<<endl;
cout<<"enter your choice:";
cin>>choice;
cout<<endl;
switch(choice){
case 1:{
int value;
int n;
cout<<"enter the total element to insert:";
cin>>n;
cout<<"enter the value to insert:";
for(int i=0;i<n;i++){
cin>>value;
root=insertNode(root,value);
}
cout<<"inorder sequence of this tree after inserting:";
inorder(root);
cout<<endl;
break;
}
case 2:{
cout<<"enter target:";
cin>>target;
if((search(root,target))==NULL){
cout<<"not found";}
else{
cout<<"found";
}
cout<<endl;
break;
}
case 3:{
cout<<"minimum node in tree is:";
findMin(root);
cout<<endl;
break;
}
case 4:{
cout<<"longest path in tree:";
cout<<lp(root);
cout<<endl;
break;
}
case 5:
{
cout<<"before swapping:";
inorder(root);
swap(root);
cout<<endl;cout<<"after swapping:";
inorder(root);
}
break;
default :
cout<<"invalid choice"<<endl;
}
} while (choice!=6);
return 0;
}
```