```
/*William Kamau M.
T00622533
---------------*/
#include<iostream>
using namespace std;
template<class T>

class bstNode{
public:
    //instance variables
    T data;
    bstNode<T>* right=NULL, * left=NULL, * parent=NULL;



};

template <class U>

class bst{
    //Instance variables

public:
 int check(bstNode<U>* n) {
        //case one: leaf
        if(n->left==NULL && n->right==NULL){
            if (n->parent->left==n) return 0;
            if (n->parent->right==n) return 1;
        }

        //case 2 one child:
        else if(n->left==NULL && n->right != NULL) return 2;

        else if(n->left != NULL && n->right ==NULL) return 3;

        //case 4 two children

        if (n->left!=NULL && n->right!=NULL) return 4;
        else return -1;
    }

    bstNode<U>* root=NULL;
    //create node
    bstNode<U>* createNode(U d){
        bstNode<U>* node=new bstNode<U>();
        node->data=d;
        return node;
    }
    //Insert method
    bstNode<U>* insert(bstNode<U>* node, U data){

        if (root==NULL){
            bstNode<U>* n=createNode(data);
            n->parent=NULL;
            root=n;
            cout<<data<<" has been added at the root"<<endl;
            return root;
        }
        else if(data>node->data&&node->right==NULL){
            bstNode<U>* n=createNode(data);
            n->parent=node;
            cout<<data<<" has been added as the right child of "<<node->data<<endl;
            node->right=n;
            return node;

        }
        else if(root!=NULL&&data<node->data&&node->left==NULL){
            bstNode<U>* n=createNode(data);
            cout<<data<<" has been added as the left child of "<<node->data<<endl;
            node->left=n;
            return node;

        }
```

```
    else if (data < node->data) node->left = insert(node->left, data);
    else if (data > node->data) node->right = insert(node->right, data);
    return node;
}
//getNode//search method
bstNode<U>* get(U e) {
    bstNode<U>* current=this->root;
    while (current!= NULL){
        if (current->data== e) return current;
        if (e>current->data) current=current->right;
        else if(e<current->data)current=current->left;
    }

    return NULL;
}


//remove method
bool remove(U e){
    bstNode<U>* n= get(e);

    if (n==NULL){

        cout<<e<<" is not in the tree";
        return false;
    }

    int i=check(n);

    switch (i){
        case 0:{
            n->parent->left=NULL;
            cout<<e<<" has been removed and the left child of "<<n->parent->data<<" has been set to NULL";
            return true;
        }
        case 1:{
            n->parent->right=NULL;
            cout<<e<<" has been removed and the right child of "<<n->parent->data<<" has been set to NULL";
            return true;

        }
        case 2:{
            cout<<"implementing case 2 removal"<<endl;
            if(n->parent->left==n){
                n->parent->left=n->right;
            }
            else if(n->parent->right==n){
                n->parent->right=n->right;
            }
            n->right->parent=n->parent;
            return true;
        }
        case 3:{
            cout<<"implementing case 3 removal"<<endl;
            if(n->parent->left==n){
                n->parent->left=n->left;
            }
            else if(n->parent->right==n){
                n->parent->right=n->left;
            }
            n->left->parent=n->parent;
            return true;
        }
        case 4:
        cout<<"Implementing case 4 removal"<<endl;

        while(n->right!=NULL){
            n->data=n->right->data;
            n=n->right;
        }
        if(n->left==NULL){
            delete n->right;
```

```
                n->right=NULL;
            }
            else{
            bstNode<U>* t=n->right;
             n->right=t->right->left;
             delete t;
             t=NULL;
            }
            return true;
        }
        return false;
    }


    void display(bstNode<U>* n){
        if (n==NULL) return;
        cout<< n->data<<", ";

        if(n->left != NULL){
            display(n->left);
        }

        if(n->right != NULL){
            display(n->right);
        }
    }

};

int main(){

    bst<int> tree;
    cout<<"Demonstrating insert function:\n\n";
    int arr[]={6,3,8,1,5,7,9,0,2,4};

    for(int i:arr){
        tree.insert(tree.root, i);
    }

    cout<<"\n\n[Display Tree]:\t";

    tree.display(tree.root);

    cout<<"\n\ndemonstrate remove function by removing the root\n\n";

    tree.remove(6);

    cout<<"\n\n[Display Tree]:\t";

    tree.display(tree.root);

    cout<<"\n\nDemonstrating search function by searchin for 0\n\n";

    bstNode<int>* n=tree.get(0);

    if (n==NULL) cout<<" not found"<<endl;
    else cout<<"found "<<n->data<<endl;




    return 0;
}
```