```cpp
#include<bits/stdc++.h>
using namespace std;
const int  MAX = 1000;

class sequential_repre{
 public:
 int  tree[MAX];
   void __init(){
   for(int i=1;i<=MAX;i++){
       tree[i]=INT_MAX;
       }
   }
   int root(int r){
       if(tree[1] !=INT_MAX)
           cout << "Root exist";
       else if(tree[1]==INT_MAX)
           tree[1] = r;
       return 0;
   }
   void Create(int loc,char side,int val){

       if(side=='L'|| side=='l')
       {
           tree[2*loc]=val;
       }
       else if(side=='R'|| side=='r')
       {
           tree[2*loc+1]=val;
       }
   }
   void Display()
   {
       for(int i=1;i<=MAX;i++){

           if(tree[i]!=INT_MAX)
           { int rightcheck=0;
           int leftcheck=0;
               cout<<tree[i] <<" Exist at -> "<<i<<"\n";
           if(tree[2*i]!=INT_MAX)
               {  leftcheck=1;
                   cout<<"Left child at location-> "<<2*i<<"\n";
               }
           if(tree[2*i+1]!=INT_MAX)
               {
               rightcheck=1;
                   cout<<" Right child at location "<<2*i+1<<"\n";
               }
               if(rightcheck==0)
               {
                   cout<<"No right child for this leaf \n";
               }
               if(leftcheck==0)
               {
                   cout<<"No left child for this leaf \n";
               }
           }
       }
       cout<<"\n";
   }
   int findindex(int tree[])
   {
       for(int i=MAX;i>=1;i--)
       {

           if(tree[i]!=INT_MAX)
           {
               return i;
           }
       }
   }

   void Delete(int pos)
   {

       if(tree[2*pos]==INT_MAX && tree[2*pos+1]==INT_MAX)
       {
           tree[pos]=INT_MAX;
       }
       else if(tree[2*pos]!=INT_MAX || tree[2*pos+1]!=INT_MAX)
       {

               int findlastindex=findindex(tree);
               tree[pos]=tree[findlastindex];
               tree[findlastindex]=INT_MAX;
       }
   }

   void Search(int v)
   {
       int check=0;
       for(int i=1;i<=MAX;i++)
       {
           if(tree[i]==v)
           {  check=1;
               cout<<"leaf present at location "<< i<<"\n";
               if(i/2!=0)
               {
                   cout<<"Its parent located at "<< i/2<<"\n";
               }
               if(i/2==0)
               {
                   cout<<"leaf itself the root \n";
               }

           }
```

```cpp
        }
        if(check==0)
        {
            cout<<"leaf with value "<<v <<"couldn't be found in entire binary tree\n";
        }

    }
};

class linked_repre{
 public:
    int INFO[MAX];
    int left[MAX];
    int right[MAX];
    int findindex1(int INFO[])
    {
        for(int i=MAX;i>=1;i--)
        {
            if(INFO[i]!=INT_MAX)
            {
                return i;
            }
        }
    }
    void __init1()
    {
        for(int i=1;i<=MAX;i++)
        {
            INFO[i]=INT_MAX;
            left[i]=INT_MAX;
            right[i]=INT_MAX;
        }
    }
    int root1(int r)
    {
        if(INFO[1] !=INT_MAX)
            cout << "Tree already had root";
        else if(INFO[1]==INT_MAX)
            INFO[1]=r;
            left[1]=2;
            right[1]=3;
        return 0;
    }
    void Create1(int loc,char side,int val)
    {
        if(side=='L'|| side=='l')
        {
            INFO[left[loc]]=val;
            left[left[loc]]=2*left[loc];
            right[left[loc]]=2*left[loc]+1;
        }
        else if(side=='R'|| side=='r')
        {
        INFO[right[loc]]=val;
        left[right[loc]]=2*right[loc];
            right[right[loc]]=2*right[loc]+1;
        }

    }
    void Display1()
    {
        for(int i=1;i<=MAX;i++)
        {  if(INFO[i]!=INT_MAX)
        {
            int rightcheck=0;
            int leftcheck=0;
                cout<<INFO[i] <<" Exist at location "<<i<<"\n";
            if(INFO[2*i]!=INT_MAX)
                {  leftcheck=1;
                    cout<<"Its left child is at location "<<2*i<<"\n";
                }
            if(INFO[2*i+1]!=INT_MAX)
                {
                rightcheck=1;
                    cout<<"Its right child is at location "<<2*i+1<<"\n";
                }
                if(rightcheck==0)
                {
                    cout<<"This leaf does not contain right child\n";
                }
                if(leftcheck==0)
                {
                    cout<<"This leaf does not contain left child\n";
                }
        }
        }
        cout<<"\n";
    }
    void Delete1(int pos)
    {
        if(right[pos]==INT_MAX && left[pos]==INT_MAX)
        {
            INFO[pos]=INT_MAX;
        }
        else if(right[pos]!=INT_MAX || left[pos]!=INT_MAX)
        {

                int findlastindex=findindex1(INFO);
                INFO[pos]=INFO[findlastindex];
                INFO[findlastindex]=INT_MAX;

        }
    }
    void Search1(int val)
    {
        int check=0;
        for(int i=1;i<=MAX;i++)
        {
            if(INFO[i]==val)
            {  check=1;
                cout<<"Present at location "<< i<<"\n";
                if(i/2!=0)
                {
                    cout<<"Its parent location is "<< i/2<<"\n";
                }
                if(i/2==0)
                {
                    cout<<"This leaf is itself the root ode of the binary tree\n";
                }

            }
        }
        if(check==0)
        {
            cout<<"leaf with value "<<val <<"couldn't be found in entire binary tree\n";
        }
    }
};

int main()
{
   int rootvalue;
   int choice;
   cout<<"Enter : \n";
   cout<<"1)By Sequential representation\n";
   cout<<"2)By Linked representation\n";
   cin>>choice;
   if(choice==1)
   {     int c;
           sequential_repre leaf;
          leaf.__init();
          int nooftime;
          cout<<"Enter the root value of tree\n";
          cin>>rootvalue;
           leaf.root(rootvalue);
            cout<<"Enter the freq. of the operatons \n";
            cin>>nooftime;
            while(nooftime--)
            {
          cout << "Enter :\n"  ;
          cout << "1)For creating a leaf in the binary tree\n";
          cout << "2)For deleting a leaf in the binary tree\n";
          cout << "3)For displaying the entire binary tree\n";
          cout << "4)For searching  leaf in the binary tree\n";
          cin>>c;
          if(c==1)

          {  int loc;
             char side;
             int val;
            cout<<"Enter the location of parent in the leaf\n";
            cin>>loc;
            cout<<"Enter L for placing as left child and R as placing as right child\n";
             cin>>side;
             cout<<"Enter the value to be inserted in the leaf\n";
            cin>>val;
            leaf.Create(loc,side,val);

          }
          else if(c==2)
          {
              int pos;
              cout<<"Enter the position of leaf to be deleted\n";
              cin>>pos;
              leaf.Delete(pos);
          }
          else if(c==3)
          {  cout<<"Binary tree in level wise is as follows:-\n";
              leaf.Display();
          }
         else if(c==4)
        {  int val;
          cout<<"Enter the value of leaf to search in the binary tree:-\n";
           cin>>val;
           leaf.Search(val);
        }
        else
        {
           cout<<"Enter correct option please\n";
        }

            }
   }
   else if(choice==2)
   {

           int c;
           linked_repre leaf1;
          leaf1.__init1();
          int nooftime;
          cout<<"Enter the root value of the binary tree\n";
          cin>>rootvalue;
           leaf1.root1(rootvalue);
            cout<<"Enter the no of times you want to perform opertion \n";
            cin>>nooftime;
            while(nooftime--)
            {
            cout<<"Enter the operation you want to perform on the binary tree\n";
          cout<<"1)For creating a leaf in the binary tree\n";
          cout<<"2)For deleting a leaf in the binary tree\n";
          cout<<"3)For displaying the entire binary tree\n";
          cout<<"4)For searching  leaf in the binary tree\n";
          cin>>c;
          if(c==1)

          {  int loc;
             char side;
             int val;
            cout<<"Enter the location of parent in the leaf\n";
            cin>>loc;
            cout<<"Enter L for placing as left child and R as placing as right child\n";
             cin>>side;
             cout<<"Enter the value to be inserted in the leaf\n";
            cin>>val;
            leaf1.Create1(loc,side,val);
          }
          else if(c==2)
          {
              int pos;
              cout<<"Enter the position of leaf to be deleted\n";
              cin>>pos;
              leaf1.Delete1(pos);
          }
          else if(c==3)
          {  cout<<"Binary tree in level wise is as follows:-\n";
              leaf1.Display1();
          }
         else if(c==4){
          int val;
          cout<<"Enter the value of leaf to search in the binary tree:-\n";
          cin>>val;
          leaf1.Search1(val);
        }
        else{
           cout<<"Enter correct option please\n";
        }
   }
   }
   else{
       cout<<"Enter correct option\n";
   }
    return 0;
}
```