

```

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

1
2
3 class sequential_repre{
4 public:
5 int tree[MAX];
6 void __init(){
7     for(int i=1;i<=MAX;i++){
8         tree[i]=INT_MAX;
9     }
10 }
11 int root(int r){
12     if(tree[1] !=INT_MAX)
13         cout << "Root exist";
14     else if(tree[1]==INT_MAX)
15         tree[1] = r;
16     return 0;
17 }
18 void Create(int loc,char side,int val){
19
20     if(side=='L' || side=='l')
21     {
22         tree[2*loc]=val;
23     }
24     else if(side=='R' || side=='r')
25     {
26         tree[2*loc+1]=val;
27     }
28 }
29 void Display()
30 {
31     for(int i=1;i<=MAX;i++){
32
33         if(tree[i]!=INT_MAX)
34         { int rightcheck=0;
35           int leftcheck=0;
36           cout<<tree[i] <<" Exist at -> " <<i<<"\n";
37           if(tree[2*i]!=INT_MAX)
38           { leftcheck=1;
39             cout<<"Left child at location-> " <<2*i<<"\n";
40           }
41           if(tree[2*i+1]!=INT_MAX)
42           {
43             rightcheck=1;
44             cout<<" Right child at location " <<2*i+1<<"\n";
45           }
46           if(rightcheck==0)
47           {
48             cout<<"No right child for this leaf \n";
49           }
50           if(leftcheck==0)
51           {
52             cout<<"No left child for this leaf \n";
53           }
54         }
55     }
56     cout<<"\n";
57 }
58 int findindex(int tree[])
59 {
60     for(int i=MAX;i>=1;i--)
61     {
62
63         if(tree[i]!=INT_MAX)
64         {
65             return i;
66         }
67     }
68 }
69
70 void Delete(int pos)
71 {
72
73     if(tree[2*pos]==INT_MAX && tree[2*pos+1]==INT_MAX)
74     {
75         tree[pos]=INT_MAX;
76     }
77     else if(tree[2*pos]!=INT_MAX || tree[2*pos+1]!=INT_MAX)
78     {
79
80         int findlastindex=findindex(tree);
81         tree[pos]=tree[findlastindex];
82         tree[findlastindex]=INT_MAX;
83     }
84 }
85
86 void Search(int v)
87 {
88     int check=0;
89     for(int i=1;i<=MAX;i++)
90     {
91         if(tree[i]==v)
92         { check=1;
93           cout<<"leaf present at location " << i<<"\n";
94           if(i/2!=0)
95           {
96             cout<<"Its parent located at " << i/2<<"\n";
97           }
98           if(i/2==0)
99           {
100             cout<<"leaf itself the root \n";
101           }
102         }
103     }
104 }

```

```

104     }
105     }
106     if(check==0)
107     {
108         cout<<"leaf with value "<<v <<"couldn't be found in entire binary tree\n";
109     }
110 }
111 }
112 };
113
114 class linked_repre{
115 public:
116     int INFO[MAX];
117     int left[MAX];
118     int right[MAX];
119     int findindex1(int INFO[])
120     {
121         for(int i=MAX;i>=1;i--)
122         {
123             if(INFO[i]!=INT_MAX)
124             {
125                 return i;
126             }
127         }
128     }
129     void __init1()
130     {
131         for(int i=1;i<=MAX;i++)
132         {
133             INFO[i]=INT_MAX;
134             left[i]=INT_MAX;
135             right[i]=INT_MAX;
136         }
137     }
138     int root1(int r)
139     {
140         if(INFO[1] !=INT_MAX)
141             cout << "Tree already had root";
142         else if(INFO[1]==INT_MAX)
143             INFO[1]=r;
144             left[1]=2;
145             right[1]=3;
146         return 0;
147     }
148     void Create1(int loc,char side,int val)
149     {
150         if(side=='L' || side=='l')
151         {
152             INFO[left[loc]]=val;
153             left[left[loc]]=2*left[loc];
154             right[left[loc]]=2*left[loc]+1;
155         }
156         else if(side=='R' || side=='r')
157         {
158             INFO[right[loc]]=val;
159             left[right[loc]]=2*right[loc];
160             right[right[loc]]=2*right[loc]+1;
161         }
162     }
163 }
164 void Display1()
165 {
166     for(int i=1;i<=MAX;i++)
167     { if(INFO[i]!=INT_MAX)
168     {
169         int rightcheck=0;
170         int leftcheck=0;
171         cout<<INFO[i] <<" Exist at location "<<i<<"\n";
172         if(INFO[2*i]!=INT_MAX)
173         { leftcheck=1;
174           cout<<"Its left child is at location "<<2*i<<"\n";
175         }
176         if(INFO[2*i+1]!=INT_MAX)
177         {
178             rightcheck=1;
179             cout<<"Its right child is at location "<<2*i+1<<"\n";
180         }
181         if(rightcheck==0)
182         {
183             cout<<"This leaf does not contain right child\n";
184         }
185         if(leftcheck==0)
186         {
187             cout<<"This leaf does not contain left child\n";
188         }
189     }
190     }
191     cout<<"\n";
192 }
193 void Delete1(int pos)
194 {
195     if(right[pos]==INT_MAX && left[pos]==INT_MAX)
196     {
197         INFO[pos]=INT_MAX;
198     }
199     else if(right[pos]!=INT_MAX || left[pos]!=INT_MAX)
200     {
201     }
202     int findlastindex=findindex1(INFO);
203     INFO[pos]=INFO[findlastindex];
204     INFO[findlastindex]=INT_MAX;
205 }
206 }
207 }
208 }
209 void Search1(int val)
210 {
211     int check=0;
212 }

```

```

211     int check=0;
212     for(int i=1;i<=MAX;i++)
213     {
214         if(INFO[i]==val)
215         { check=1;
216           cout<<"Present at location "<<i<<"\n";
217           if(i/2!=0)
218           {
219               cout<<"Its parent location is "<<i/2<<"\n";
220           }
221           if(i/2==0)
222           {
223               cout<<"This leaf is itself the root ode of the binary tree\n";
224           }
225         }
226     }
227 }
228 if(check==0)
229 {
230     cout<<"leaf with value "<<val <<"couldn't be found in entire binary tree\n";
231 }
232 }
233 };
234
235
236 int main()
237 {
238     int rootvalue;
239     int choice;
240     cout<<"Enter : \n";
241     cout<<"1)By Sequential representation\n";
242     cout<<"2)By Linked representation\n";
243     cin>>choice;
244     if(choice==1)
245     { int c;
246       sequential_repre leaf;
247       leaf.__init();
248       int nooftime;
249       cout<<"Enter the root value of tree\n";
250       cin>>rootvalue;
251       leaf.root(rootvalue);
252       cout<<"Enter the freq. of the operatons \n";
253       cin>>nooftime;
254       while(nooftime--)
255       {
256           cout << "Enter : \n" ;
257           cout << "1)For creating a leaf in the binary tree\n";
258           cout << "2)For deleting a leaf in the binary tree\n";
259           cout << "3)For displaying the entire binary tree\n";
260           cout << "4)For searching leaf in the binary tree\n";
261           cin>>c;
262           if(c==1)
263
264           { int loc;
265             char side;
266             int val;
267             cout<<"Enter the location of parent in the leaf\n";
268             cin>>loc;
269             cout<<"Enter L for placing as left child and R as placing as right child\n";
270             cin>>side;
271             cout<<"Enter the value to be inserted in the leaf\n";
272             cin>>val;
273             leaf.Create(loc,side,val);
274
275           }
276           else if(c==2)
277           {
278               int pos;
279               cout<<"Enter the position of leaf to be deleted\n";
280               cin>>pos;
281               leaf.Delete(pos);
282           }
283           else if(c==3)
284           { cout<<"Binary tree in level wise is as follows:-\n";
285             leaf.Display();
286           }
287           else if(c==4)
288           { int val;
289             cout<<"Enter the value of leaf to search in the binary tree:-\n";
290             cin>>val;
291             leaf.Search(val);
292           }
293           else
294           {
295               cout<<"Enter correct option please\n";
296           }
297       }
298   }
299 }
300 else if(choice==2)
301 {
302
303     int c;
304     linked_repre leaf1;
305     leaf1.__init1();
306     int nooftime;
307     cout<<"Enter the root value of the binary tree\n";
308     cin>>rootvalue;
309     leaf1.root1(rootvalue);
310     cout<<"Enter the no of times you want to perform opertion \n";
311     cin>>nooftime;
312     while(nooftime--)
313     {
314         cout<<"Enter the operation you want to perform on the binary tree\n";
315         cout<<"1)For creating a leaf in the binary tree\n";
316         cout<<"2)For deleting a leaf in the binary tree\n";
317         cout<<"3)For displaying the entire binary tree\n";

```

```
319 cout<<"4)For searching leaf in the binary tree\n";
320 cin>>c;
321 if(c==1)
322
323 { int loc;
324   char side;
325   int val;
326   cout<<"Enter the location of parent in the leaf\n";
327   cin>>loc;
328   cout<<"Enter L for placing as left child and R as placing as right child\n";
329   cin>>side;
330   cout<<"Enter the value to be inserted in the leaf\n";
331   cin>>val;
332   leaf1.Create1(loc,side,val);
333 }
334 else if(c==2)
335 {
336   int pos;
337   cout<<"Enter the position of leaf to be deleted\n";
338   cin>>pos;
339   leaf1.Delete1(pos);
340 }
341 else if(c==3)
342 { cout<<"Binary tree in level wise is as follows:-\n";
343   leaf1.Display1();
344 }
345 else if(c==4){
346   int val;
347   cout<<"Enter the value of leaf to search in the binary tree:-\n";
348   cin>>val;
349   leaf1.Search1(val);
350 }
351 else{
352   cout<<"Enter correct option please\n";
353 }
354 }
355 }
356 else{
357   cout<<"Enter correct option\n";
358 }
359 return 0;
360 }
```