

```

1  #include <iostream>
2  #include <math.h>
3  using namespace std;
4
5  int pow2(int x);
6  void tree update(int index, int value);
7  int tree query(int left, int right);
8  void tree build();
9  int tree get leaf(int x);
10 int tree get left child(int x);
11 int tree get right child(int x);
12 int tree get parent(int x);
13 void tree print();
14 bool is left(int x);
15 bool is right(int x);
16 void tree update(int index, int value);
17
18 int n;
19 const int MAXN = 100000; ///100 000
20 int a[MAXN];
21 const int MAXA = 100000; ///100 000
22
23 const int TREE MAX = ceil(log2(MAXA));
24 int tree[1 << TREE MAX];
25 int tree height;
26
27 void tree update(int index, int value) {
28     index = tree get leaf(index);
29     tree[index] = value;
30     index = tree get parent(index);
31     while(index >= 0) {
32         tree[index] = max(tree[tree get left child(
index)], tree[tree get right child(index)]);
33         index = tree get parent(index);
34     }
35 }
36
37 int tree query(int left, int right) {
38     left = tree get leaf(left);
39     right = tree get leaf(right);
40     int ans = max(tree[left], tree[right]);
41     while(left + 1 < right) {
42         if(is left(left)) {
43             ans = max(ans, tree[left + 1]);
44         }
45         if(is right(right)) {
46             ans = max(ans, tree[right - 1]);
47         }
48         left = tree get parent(left);
49         right = tree get parent(right);
50     }
51     return ans;
52 }
53
54 void tree build() {
55     tree height = ceil(log2(n)) + 1;
56     for(int i = 0; i < n; ++i) {
57         tree[tree get leaf(i)] = a[i];
58     }
59     for(int i = tree get leaf(0) - 1; i >= 0; --i)
60         tree[i] = max(tree[tree get left child(i)],
tree[tree get right child(i)]);
61     tree print();
62 }
63

```

```

64
65 void tree print() {
66     int lvl = 1;
67     int lvl curr = 0;
68     for(int i = 0; i <= tree get leaf(n - 1); ++i)
69         cout << tree[i] << " ";
70         lvl curr++;
71         if(lvl curr >= lvl) {
72             lvl curr = 0;
73             cout << endl;
74             lvl *= 2;
75         }
76     }
77 }
78
79 int tree get leaf(int x) {
80     return x + pow2(tree height - 1) - 1;
81 }
82
83 int tree get left child(int x) {
84     return x * 2 + 1;
85 }
86
87 int tree get right child(int x) {
88     return tree get left child(x) + 1;
89 }
90
91 int pow2(int x) {
92     return 1 << x;
93 }
94
95 int tree get parent(int x) {
96     if(is left(x)) {
97         x += 1;
98     }
99     x /= 2;
100    x--;
101    return x;
102 }
103
104 bool is left(int x) {
105     return x % 2 == 1;
106 }
107
108 bool is right(int x) {
109     return !is left(x);
110 }
111
112 int main() {
113     cin >> n;
114     for(int i = 0; i < n; ++i) {
115         cin >> a[i];
116     }
117     tree build();
118     string type = "";
119     int arq1, arq2;
120     while(cin >> type) {
121         if(type == "u" || type == "update") {
122             cin >> arq1 >> arq2;
123             tree update(arq1, arq2);
124         }
125         if(type == "q" || type == "query") {
126             cin >> arq1 >> arq2;
127             cout << tree query(arq1, arq2) << endl;
128         }

```

```
129         if (type == "p" || type == "print") {
130             tree_print();
131         }
132     }
133     return 0;
134 }
```