```cpp
#include <iostream>
#include <math.h>
using namespace std;

int pow2(int x);
void tree_update(int index, int value);
int tree_query(int left, int right);
void tree_build();
int tree_get_leaf(int x);
int tree_get_left_child(int x);
int tree_get_right_child(int x);
int tree_get_parent(int x);
void tree_print();
bool is_left(int x);
bool is_right(int x);
void tree_update(int index, int value);

int n;
const int MAXN = 100000;///100 000
int a[MAXN];
const int MAXA = 100000;///100 000

const int TREE_MAX = ceil(log2(MAXA));
int tree[1 << TREE_MAX];
int tree_height;

void tree_update(int index, int value) {
    index = tree_get_leaf(index);
    tree[index] = value;
    index = tree_get_parent(index);
    while(index >= 0) {
        tree[index] = max(tree[tree_get_left_child(index)], tree[tree_get_right_child(index)]);
        index = tree_get_parent(index);
    }
}

int tree_query(int left, int right) {
    left = tree_get_leaf(left);
    right = tree_get_leaf(right);
    int ans = max(tree[left], tree[right]);
    while(left + 1 < right) {
        if(is_left(left)) {
            ans = max(ans, tree[left + 1]);
        }
        if(is_right(right)) {
            ans = max(ans, tree[right - 1]);
        }
        left = tree_get_parent(left);
        right = tree_get_parent(right);
    }
    return ans;
}

void tree_build() {
    tree_height = ceil(log2(n)) + 1;
    for(int i = 0; i < n; ++i) {
        tree[tree_get_leaf(i)] = a[i];
    }
    for(int i = tree_get_leaf(0) - 1; i >= 0; --i) {
        tree[i] = max(tree[tree_get_left_child(i)], tree[tree_get_right_child(i)]);
    }
    tree_print();
}
```

```
void tree_print() {
    int lvl = 1;
    int lvl_curr = 0;
    for(int i = 0; i <= tree_get_leaf(n - 1); ++i) {
        cout << tree[i] << " ";
        lvl_curr++;
        if(lvl_curr >= lvl) {
            lvl_curr = 0;
            cout << endl;
            lvl *= 2;
        }
    }
}

int tree_get_leaf(int x) {
    return x + pow2(tree_height - 1) - 1;
}

int tree_get_left_child(int x) {
    return x * 2 + 1;
}

int tree_get_right_child(int x) {
    return tree_get_left_child(x) + 1;
}

int pow2(int x) {
    return 1 << x;
}

int tree_get_parent(int x) {
    if(is_left(x)) {
        x += 1;
    }
    x /= 2;
    x--;
    return x;
}

bool is_left(int x) {
    return x % 2 == 1;
}

bool is_right(int x) {
    return !is_left(x);
}

int main() {
    cin >> n;
    for(int i = 0; i < n; ++i) {
        cin >> a[i];
    }
    tree_build();
    string type = "";
    int arg1, arg2;
    while(cin >> type) {
        if(type == "u" || type == "update") {
            cin >> arg1 >> arg2;
            tree_update(arg1, arg2);
        }
        if(type == "q" || type == "query") {
            cin >> arg1 >> arg2;
            cout << tree_query(arg1, arg2) << endl;
        }
        if(type == "p" || type == "print") {
            tree_print();
        }
    }
    return 0;
}
```