

Problem

Result

Code : Max Priority Queue

Send Feedback

Implement the class for Max Priority Queue which includes following functions -

- getSize** -
Return the size of priority queue i.e. number of elements present in the priority queue.
- isEmpty** -
Check if priority queue is empty or not. Return true or false accordingly.
- insert** -
Given an element, insert that element in the priority queue at the correct position.
- getMax** -
Return the maximum element present in the priority queue without deleting. Return -Infinity if priority queue is empty.
- removeMax** -
Delete and return the maximum element present in the priority queue. Return -Infinity if priority queue is empty.

Note : main function is given for your reference which we are using internally to test the class.

```

1
2 //*****
3 * Following is the main function that er are using internally
4 *
5
6 int main() {
7     PriorityQueue pq;
8     int choice;
9     cin >> choice;
10    while(choice != -1) {
11        switch(choice) {
12            case 1 : // insert
13                int element;
14                cin >> element;
15                pq.insert(element);
16                break;
17            case 2 : // getMax
18                cout << pq.getMax() << endl;
19                break;
20            case 3 : // removeMax
21                cout << pq.removeMax() << endl;
22                break;
23            case 4 : // size
24                cout << pq.getSize() << endl;
25                break;
26            case 5 : // isEmpty
27                if(pq.isEmpty()) {
28                    cout << "true" << endl;
29                }
30                else {
31                    cout << "false" << endl;
32                }
33            default :
34                return 0;
35        }
36        cin >> choice;
37    }
38 }

```

PREVIOUS

NEXT

CUSTOM INPUT

SUBMIT SOLUTION

```

41 class PriorityQueue {
42 // Complete this class
43 vector<int> pq;
44
45 public:
46 PriorityQueue(){
47
48 }
49
50 int getSize(){
51     return pq.size();
52 }
53
54 bool isEmpty(){
55     return pq.size()==0;
56 }
57
58 int getMax(){
59     if(isEmpty()){
60         return 0;
61     }
62     return pq[0];
63 }
64
65 void insert(int element){
66     pq.push_back(element);
67
68     int childIndex = pq.size() - 1;
69
70     while(childIndex > 0) {
71         int parentIndex = (childIndex - 1) / 2;
72

```

CUSTOM INPUT


SUBMIT SOLUTION

```

76     pq[parentIndex] = temp;
77 }
78 else {
79     break;
80 }
81     childIndex = parentIndex;
82 }
83
84 }
85
86
87 int removeMax() {
88
89     int ans = pq[0];
90     int last = pq.size()-1;
91
92     int temp = pq[0];
93     pq[0] = pq[last];
94     pq[last] = temp;
95
96     pq.pop_back();
97
98     int parent = 0;
99     int childRight = 2*parent+2;
100    int childLeft = 2*parent+1;
101
102    while(childLeft<pq.size()){
103
104
105    if( childRight<pq.size()-1 &&pq[parent]<max(pq[childLeft],pq[childRight])){
106        if(pq[childRight]>pq[childLeft]){
107            int temp = pq[childRight];
108            pq[childRight] = pq[parent];
109            pq[parent] = temp;
110
111            parent = childRight;
112        }else{
113            int temp = pq[childLeft];
114            pq[childLeft] = pq[parent];
115            pq[parent] = temp;
116            parent = childLeft;
117        }

```

```
118     }
119     else if(pq[parent]<pq[childLeft]){
120         int temp = pq[childLeft];
121         pq[childLeft] = pq[parent];
122         pq[parent] = temp;
123         parent = childLeft;
124     }
125     else{
126         break;
127     }
128     childRight = 2*parent+2;
129     childLeft = 2*parent+1;
130 }
131 return ans;
132 }
133
134
135
136
137 };
```

 CUSTOM INPUT

 SUBMIT SOLUTION

