```cpp
/*
Filename: p6.cpp
Author(s): Zachary Rea and Parker Ross
Date: 26 February 2023
Description: The cpp for Priority Queue
*/
#include <iostream>  //allows for usage of cin, cout, and cerr

#include "p6.h"

using namespace std;
#include <cmath>
//****************************************************************************
//Constructors and De-constructors
//****************************************************************************
//Constructor
//Written by Parker

iPQ::iPQ(int n) {
    qCount = 0;
    qCapacity = n;
    values = new int[n];
}

//****************************************************************************
//de-constructor
//Written by Parker

iPQ::~iPQ() {
    delete[] values;
}

//****************************************************************************
//Private Members
//Written by Zach

int iPQ::parent(int index) const{
    int rc;
    if (index) {
        rc = (index-1)/2;
    } else {
        rc = 0;
    }
    return rc;
}

//****************************************************************************
//Function for finding left child
//Written by Zach

int iPQ::left(int index) const{
    return (2*index)+1;
}

//****************************************************************************
//Function for finding right child
//Written by Zach

int iPQ::right(int index) const{
    return (2*index)+2;
}
//****************************************************************************
//function to print tree by level
//written by Parker

void iPQ::printIt(int ind, int count) const{
    int start, stop;
    count = 1 << ind;
    start = count - 1;
```

```
70          stop = start + count;
71          if(start < qCount){
72              if(stop > qCount){
73                  stop = qCount;
74              }
75              cout << "Level[" << ind << "]-> ";
76              for(int i = start; i < stop; i++){
77                  cout << values[i] << " ";
78              }
79              cout << endl;
80              printIt(ind + 1,count);
81              ind++;
82          }
83      }
84
85      //*****************************************************************************
86      //Function to swap two integers with each other
87      //Written by Zach
88
89      void iPQ::swap(int x, int y) {
90      int t = values[x];
91      values[x] = values[y];
92      values[y] = t;
93      }
94
95      //*****************************************************************************
96      //Function to perform heap Bubble-Up operation
97      //written by Parker
98
99      void iPQ::bubbleUp(int index) {
100         if (index) {
101             int par = parent(index);
102             if(values[index] > values[par]) {
103                 swap(index,par);
104                 bubbleUp(par);
105             }
106         }
107     }
108
109     //*****************************************************************************
110     //Function to perform heapify operation
111     //Written by Zach
112
113     void iPQ::heapify(int index) {
114         if (index >= 0) {
115             int larger = index;
116             int l = left(index);
117             if (l < qCount) {
118                 //check if left is larger
119                 if (values[l] > values[larger]) {
120                     larger = l;
121                 }
122                 int r = right(index);
123                 if (r < qCount) {
124                     //check if right is larger
125                     if (values[r] > values[larger]) {
126                         larger = r;
127                     }
128                 }
129             }
130             //swap if needed
131             if (index != larger) {
132                 swap(index,larger);
133                 heapify(larger);
134             }
135         }
136     }
137
138     //*****************************************************************************
```

```
139    //Public Members
140    //********************************************************************
141    //Enques value into iPQ; returns true for success; false if full
142    //Written by Parker
143
144    bool iPQ::enq(int v) {
145        bool rc = false;
146        if(qCount < qCapacity){
147            values[qCount] = v;
148            qCount++;
149            bubbleUp(qCount - 1);
150            rc = true;
151        }
152        return rc;
153    }
154
155    //********************************************************************
156    //returns true if IPQ is not empty; removes & returns max value, false if empty
157    //Written by Parker
158
159    bool iPQ::deq(int &v) {
160        bool rc = qCount > 0;
161        if (rc){
162            v = values[0];
163            qCount--;
164            values[0] = values[qCount];
165            heapify(0);
166        }
167        return rc;
168    }
169
170    //********************************************************************
171    //Function for printing the array
172    //Written by Zach
173
174    void iPQ::printIt() const{
175        printIt(0,0);
176    }
177
178    //********************************************************************
179    //Function for clearing the contents of the array
180    //Written by Zach
181
182    void iPQ::clear() {
183        qCount = 0;
184    }
185
186    //********************************************************************
187    //Function for returning the size of the array
188    //Written by Zach
189
190    int iPQ::count() const{
191        return qCount;
192    }
```