#include <iostream>

using namespace std;

/\* PriorityQueue class implementation using an array.

The node count is stored in the first index of the queue array. \*/

class PriorityQueue

{

private:

unsigned int const size = pow(10, 3);

unsigned int\* queue;

public:

PriorityQueue()

{

queue = new unsigned int[size];

for (unsigned int i = 0; i < size; i++) queue[i] = 0; // initialize to 0

}

/\* Insert a value into the queue and bubble up as needed \*/

void Insert(int key)

{

int currentPosition = ++queue[0];

queue[currentPosition] = key;

while (currentPosition > 1 && queue[currentPosition] < queue[(currentPosition/2)])

{

int temp = queue[currentPosition];

// swap parent and child values

queue[currentPosition] = queue[(currentPosition / 2)];

queue[(currentPosition / 2)] = temp;

currentPosition = (currentPosition / 2);

}

}

/\* Extract root of array and bubble down as needed \*/

int ExtractMin()

{

// pull root out

int min = queue[1];

// move right most leaf to root

queue[1] = queue[queue[0]];

queue[queue[0]--] = 0; // set right most leaf node to 0 (and decrease queue size)

int currentPosition = 1;

int leftChild = currentPosition \* 2;

int rightChild = ((double)currentPosition + (double).5) \* 2;

while (currentPosition < queue[0] && (queue[currentPosition] > queue[leftChild] || queue[currentPosition] > queue[rightChild]))

{

leftChild = currentPosition \* 2;

rightChild = ((double)currentPosition + (double).5) \* 2;

if (queue[leftChild] < queue[rightChild]) // bubble down left edge

{

int temp = queue[currentPosition];

// swap parent and child values

queue[currentPosition] = queue[leftChild];

queue[leftChild] = temp;

currentPosition = leftChild;

}

else if (queue[rightChild] <= queue[leftChild]) // right edge

{

int temp = queue[currentPosition];

// swap parent and child values

queue[currentPosition] = queue[rightChild];

queue[rightChild] = temp;

currentPosition = rightChild;

}

}

return min;

}

// Print each level of a tree with each node separated by spaces

// a b-tree has 2^(n)/2 leaf nodes at a given height n

void PrintQueue()

{

unsigned int height = log2(queue[0])+1;

unsigned int leafCount = pow(2, height) / 2;

cout << "Length: " << queue[0] << endl;

cout << "Tree Height: " << height << endl;

cout << "Leaf count: " << leafCount << endl;

int t = 1;

for (unsigned int i = 1; i <= height; i++)

{

for (unsigned int j = 1; j <= (pow(2, i) / 2); j++)

{

if (queue[t] == 0)

{

cout << " ";

t++;

}

else

cout << queue[t++] << " ";

}

cout << endl;

}

cout << endl;

}

};