Updating Keys in a Binary Heap

In Lecture, we went through a Binary Heap implementation that supports the operations **Insert**() and **ExtractMax**(). You are now asked to extend this implementation to support a new operation. **UpdateKey**(). This operation will update the value of one element in the Binary Heap. In summary, you are asked to implement a Binary Heap that can support the following operations:

Operation	**Description**
`I [x]`	Insert **[x]** into the Binary Heap.
E	Extract the maximum element from the Binary Heap and output it. It is guaranteed there is at least one element in the Binary Heap when this operation is run.
`U [x] [y]`	Update **one copy** of **[x]** to **[y]**. It is guaranteed there is at least one copy of **[x]** in the Binary Heap when this operation is run.

All the operations should run in O(log **N**) time where **N** is the number of elements in the Binary Heap. In addition, print all the elements left in the Binary Heap at the end of all the operations in **non-increasing** order. It is guaranteed there will be at least one element in the Binary Heap at the end of all the operations.

Input
The first line of input contains an integer **Q**. **Q** lines will follow, representing an operation each. The operations should be executed in order and the format would be as described in the table above. (See sample)

Output
For each **E** operation, output the element extracted.
At the end of all **Q** operations, output all the elements left in the Binary Heap at the end of all the operations in **non-increasing** order. Add a single space between two consecutive values. **Do not print a space after the last value.** Instead, remember to print an end-line character at the end of the output.

Limits
- $1 \le$ **Q** $\le 200{,}000$
- All the values will range from 1 to 10^9 inclusive.
- It is guaranteed there is at least one element in the Binary Heap when an **E** operation happens.
- It is guaranteed there will be at least one element in the Binary Heap at the end of all the operations.

Sample Input (**updatepq1.in**)	Sample Output (**updatepq1.out**)

7
I 5
I 4
I 4
I 5
U 4 6
E
E
	6
	5
	5 4

Explanation

After the first 4 **I** operations, the priority queue will contain [5, 5, 4, 4] in non-increasing order. Then, the **U** operation updates one copy of 4 to 6, giving [6, 5, 5, 4]. After that, top 2 elements are extracted which are 6 and 5 respectively, leaving [5, 4] in the priority queue at the end of all the operations.

Notes:

1. **You should develop your program by extending the `BinaryHeap` class given in lecture.** This has been provided to you as `BinaryHeap.java` that has an additional empty `UpdateKey()` method that you should implement.
2. You are not allowed to import any additional Java classes other than **`HashSet/HashMap.`**
3. You may add any helper classes or methods as needed. You will also likely need to modify the implementations of the other methods to support this new method. However, you should not modify the function signatures of **Insert, UpdateKey and ExtractMax.**
4. You should **NOT** modify `Grader.java` and you should submit only `BinaryHeap.java` to CodeCrunch
5. To test your code, you can run **`bash check.sh Grader`**