Priority Queue System

Imagine you are developing a priority queue system for a dynamic scheduling application where tasks need to be processed based on their priority. A task is defined by a pair (**task_id**, **priority**). You decide to use a Heap (H) for this purpose, which allows efficient retrieval of the task with the highest priority in time $O(1)$. You are assigned to perform the following:

(a) **InsertTask(H, task)**: Insert the given task with unique `task_id` and unique `priority` (both positive integers $\in [1, 10^6]$) into the Heap H. After insertion, perform the necessary operations to maintain the heap property.

(b) **UpdatePriorityValue(H, `task_id`, `newValue`)**: Update the priority of the given task with `task_id` to the `newValue`. Perform the necessary operations to maintain the heap property.

(c) **DeleteMax(H)**: Delete the task with maximum priority in the heap H and print the corresponding `task_id`. After deletion, perform the necessary operations to maintain the heap property.

(d) **DisplayHeap(H)**: Display the current state of the heap H as a space-separated list of `task_ids` in level-order traversal.

(e) **CheckCousins(H, `task_id1`, `task_id2`)**: Given two `task_ids` in the heap with `task_id1` and `task_id2`, check if they are cousins in the binary tree corresponding to the heap H. A pair of nodes in a binary tree are cousins if they are at the same level but have different parents.

Assume that all task_ids and priorities are unique at all times.

Input Format:

Each line contains a character from {'i', 'u', 'd', 's', 'c', 'e'} followed by zero or more positive integers.

- Character `'i'` is followed by two positive integers: `task_id` and `priority`. Perform the **InsertTask(H, task)** operation.
- Character `'u'` is followed by two positive integers `task_id` and `newValue`. Perform the **UpdatePriorityValue(H, `task_id`, `newValue`)** operation.
- Character `'d'` performs the **DeleteMax(H)** operation.
- Character `'s'` performs the **DisplayHeap(H)** operation.
- Character `'c'` is followed by two positive integers `task_id1` and `task_id2`. Perform the **CheckCousins(H, `task_id1`, `task_id2`)** operation.
- Character `'e'` is to terminate the sequence of operations.

Output Format:

The output (if any) of each command should be printed on a separate line. However, no output is printed for `'i'` and `'e'`.

- For Option `'u'`: Print `-1` if the node is not found else print the `task_id`.
- For Option `'d'`: Print the `task_id` of the node with maximum priority extracted from the heap H. If heap is empty print `-1`.
- For Option `'s'`: Print the Heap as a space-separated list of `task_ids` in level-order traversal. If the heap is empty, print `-1`.
- For Option `'c'`: If the two nodes are cousins, print `yes` else `no` (small letters).

Sample Test Cases

Input 1:

```
i 21 100
i 22 95
i 23 85
i 24 75
i 25 65
i 26 60
d
s
u 23 125
s
c 22 23
i 35 200
s
e
```

Output 1:

```
21
22 24 23 26 25
23
23 24 22 26 25
no
35 24 23 26 25 22
```

Input 2:

```
i 71 80
```

```
i 72 90
i 73 70
i 74 60
i 75 85
i 76 100
s
c 74 73
d
u 74 95
s
d
c 71 72
e
```

Output 2:

```
76 75 72 74 71 73
yes
76
74
74 72 73 75 71
74
no
```