Question 1 (Afternoon Lab)

Implement a Binary min-Heap using an array. Each element of the array should be a pointer to the following structure.

```
typedef struct _Element {
      char first_name[100];
      char last_name[100];
      int heap_index;
} Element;
```

where first_name and last_name are two strings and the heap_index holds the current position of an element in the array.
A node will be minimum if its first_name is lexicographically smaller than that of all other nodes. In case of first_name being equal, then comparison should be done on the basis of last_name (lexicographically smaller last_name would be min)

Implement the following functions:

- **InitHeap (char* first_name, char* last_name):** Creates a heap with a single element
- **Insert (char *first_name, char* last_name):** Inserts an element into the heap
- **FindMin():** Returns the top element of the heap
- **DeleteMin ():** Deletes the top element of the heap
- **Delete (int index):** Delete the element in indexed position, if it is there

Input/Output
The first line of the input contains T, the number of heap operations to be performed.
-> InitHeap -
    Followed by a pair of strings (Space separated, First one to initiate the heap).
    No output required.
    First operation will always be InitHeap, and it'd never repeat in input file.
-> Insert -
    Followed by the pair of strings (space separated) to be added to its appropriate position.
    Print the position where the element was inserted in the minHeap.
-> FindMin -
    Prints the top minimum pair of string (first_name last_name).
-> DeleteMin -
    Deletes the top pair of string
    Print -1 if the heap is empty.
    Otherwise, print the pair of strings at the top node, and delete it.
-> Delete
    Followed by an integer specifying the index of the node to be deleted.

Print -1 if no node exists at the specified index.
Otherwise print the pair of strings (space-separated) at the given node, and delete it.

Constraints
1 ≤ Length(Strings) ≤ 100
1 ≤ Number of Nodes ≤ 10^6

Sample Input
```
8
InitHeap abc def
Insert aaa feg
Insert bcd bhg
DeleteMin
FindMin
FindMin
DeleteMin
Delete 2
```

Sample Output
```
1
3
aaa feg
abc def
abc def
abc def
-1
```

Explanation
#1 Heap: {1: (abc,def)}
#2 Heap: {1: (aaa,feg), 2: (abc,def)}
#3 Heap: {1: (aaa,feg), 2: (abc,def), 3:(bcd,bhg)}
#4 Heap: {1: (abc,def), 2:(bcd,bhg)}
#5 Heap: {1: (abc,def), 2:(bcd,bhg)}
#6 Heap: {1: (abc,def), 2:(bcd,bhg)}
#7 Heap: {1:(bcd,bhg)}
#8 Heap: {1:(bcd,bhg)}