Minimal Heap

This purpose of this project is to make a minimal heap according to the priority value.

There are many events. Each event has a priority value (in the range of 1~100).

The lower value means the larger priority. That is, the value *“1”* is the maximal and value *“100”* is the minimal. The event we have to extract must have the maximal priority value. Intuitive meaning is that build the minimal heap according the priority value, so each time the event you extract has the maximal priority.

The procedures you have to write are listed below:

**1. Minimal(A)** : **only** output the minimal event

**2. Extract-min(A)**: extract the minimal event and then adjust the heap to not violate the heap property

**3. Insert(A,key)**: insert the event with priority value “key”. Similarly , you have to maintain the heap property

**4. Decrease-key(A, I, key)** : The priority value of A[i] has to decrease to “Key”. Similarly, you have to maintain the heap property

**Sample Input:**

1 100 ;event ID , priority value

2 50

3 45

4 60

5 70

\* ; Operation are listed below

I ;insert event 1

I ; insert event 2

I ; insert event 3

E ; extract the minimal event

E ;extract the minimal event

I ;insert the event 4

I ;insert the event 5

D(1,99) ;decrease(A ,1,99)

M ;output the minimal

E ; extract the minimal event

M ; output the minimal event

Firstly, there are sequences of event with the priority value. Each event is assigned a event ID in a serial order.

Below the symbol ‘\*’, we will give the input operations:

(**M**:minimal, **I**:insert, **E**:extract, **D**(I,k):decrease(A,I,k) )

Sample **Output :**

100

50 100

45 100 50

3

2

60 100

60 100 70

70 100 99

70

5

4

(1)After each **Insert** ,**Decrease** operation, you have to output the priority value of A in the order of index ( A[1],A[2],….).

(2)After each **Minimal** and **Extract-min** operation, output the eventId of this operation