Min heap

→ Build Min heap :-

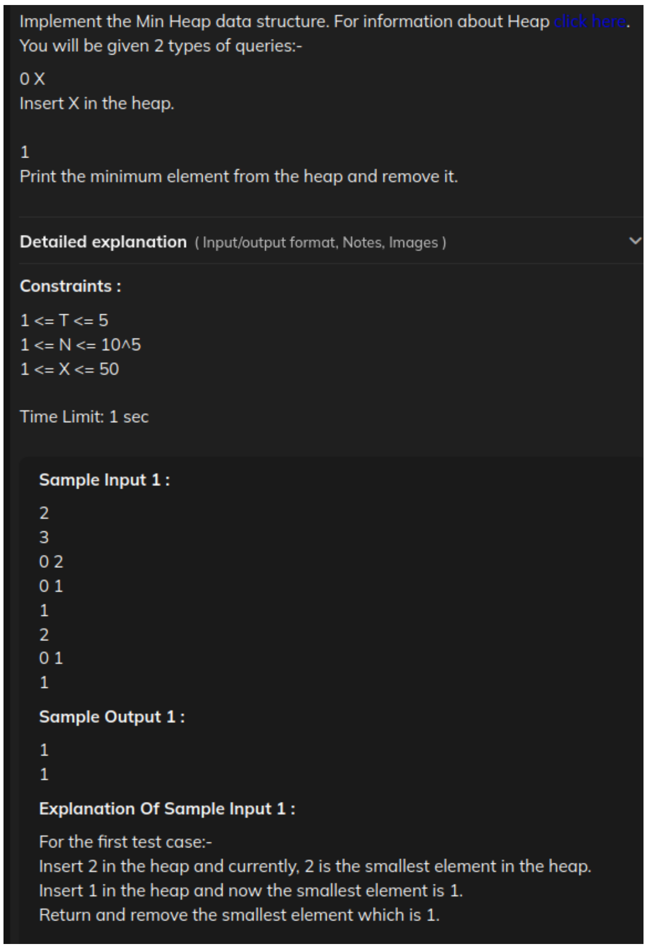
```
#include <bits/stdc++.h>

void heapify(vector<int> &arr, int i, int n) {
    int smallest = i;

    int leftChild = 2 * i + 1;
    int rightChild = 2 * i + 2;
    if(leftChild < n && arr[leftChild] < arr[smallest]) smallest = leftChild;
    if(rightChild < n && arr[rightChild] < arr[smallest]) smallest = rightChild;

    if(smallest ≠ i) {
        swap(arr[i], arr[smallest]);
        heapify(arr, smallest, n);
    }
}

vector<int> buildMinHeap(vector<int> &arr)
{
    int n = arr.size();
    for(int i = n / 2 - 1; i ≥ 0; i--) {
        heapify(arr, i, n);
    }
    return arr;
}
```



Implement the Min Heap data structure. For information about Heap click here.
You will be given 2 types of queries:-

0 X
Insert X in the heap.

1
Print the minimum element from the heap and remove it.

Detailed explanation (Input/output format, Notes, Images)

Constraints :

1 <= T <= 5
1 <= N <= 10^5
1 <= X <= 50

Time Limit: 1 sec

Sample Input 1 :

```
2
3
0 2
0 1
1
2
0 1
1
```

Sample Output 1 :

```
1
1
```

Explanation Of Sample Input 1 :

For the first test case:-
Insert 2 in the heap and currently, 2 is the smallest element in the heap.
Insert 1 in the heap and now the smallest element is 1.
Return and remove the smallest element which is 1.

```
void heapify(vector<int> &heap, int i, int n) {
  int smallest = i;

  int leftChild = 2 * i + 1;
  int rightChild = 2 * i + 2;
  if (leftChild < n && heap[leftChild] < heap[smallest])
    smallest = leftChild;
  if (rightChild < n && heap[rightChild] < heap[smallest])
    smallest = rightChild;

  if (smallest ≠ i) {
    swap(heap[i], heap[smallest]);
    heapify(heap, smallest, n);
  }
}

void deleteHeap(vector<int> &heap) {
  if (heap.size() == 0)
    return;
  int size = heap.size();
  if (size == 1) {
    heap.erase(heap.begin());
    return;
  }
  heap[0] = heap[size - 1];
  heap.pop_back();
  heapify(heap, 0, size);
}

void insert(vector<int> &heap, int val) {
  heap.push_back(val);
  int index = heap.size() - 1;

  while (index > 0) {
    int parent = (index - 1) >> 1;
    if (heap[parent] > heap[index])
      swap(heap[parent], heap[index]);
    index = parent;
  }
}

vector<int> minHeap(int n, vector<vector<int>> &q) {
  vector<int> ans;
  vector<int> heap;
  for (int i = 0; i < q.size(); i++) {
    if (q[i].size() > 1) {
      int val = q[i][1];
      insert(heap, val);
    } else {
      ans.push_back(heap[0]);
      deleteHeap(heap);
    }
  }
  return ans;
}
```