Algorithms about heap:

(1) Sort:

```
while (size > 1)
{
    swap(heap, 1, size)
    size--;
    siftDown(heap, 1)
}
```

(2) delete / dequeue (serve):

```
e = heap[1]
heap[1] = heap[size]
size--
siftDown(heap, 1)
```

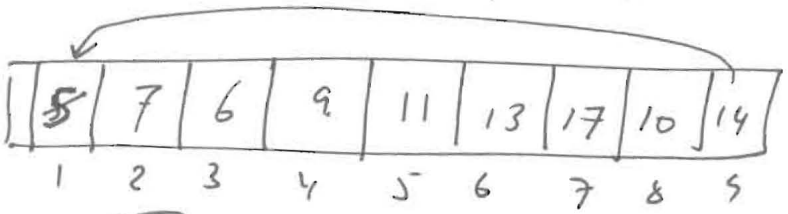
(3) insert / enqueue:

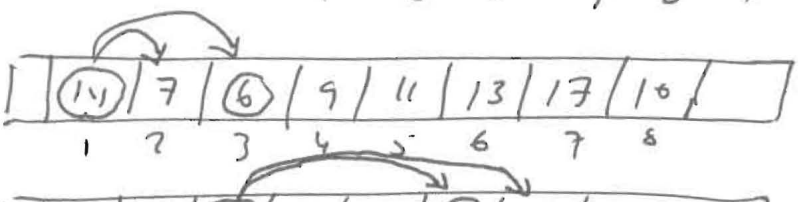
```
size = size + 1
heap[size] = e
siftup(heap, size)
```

```
public boolean isHeap(int x[], int size)
{
    int i = 1;
    while (x[i] < x[2*i] && x[i] < x[2*i+1] &&
           i < size/2)
        i++;
    if (i <= size/2)
        return false
    else
        return true;
}
```

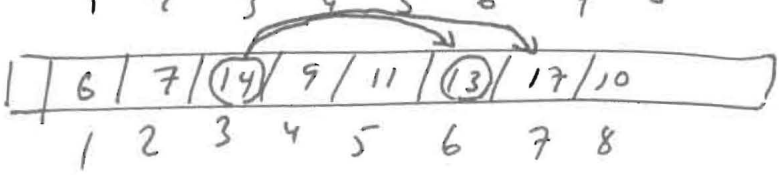
Q4 : priority Queue as heap, lower value higher p

Serve one time (dequeue)

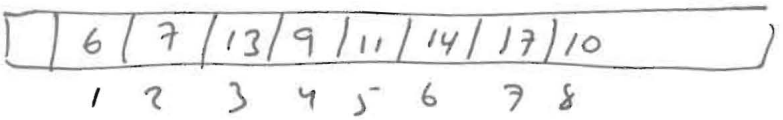




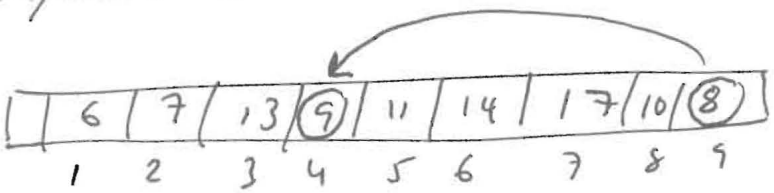
i = 1
Left = 2
Right = 3



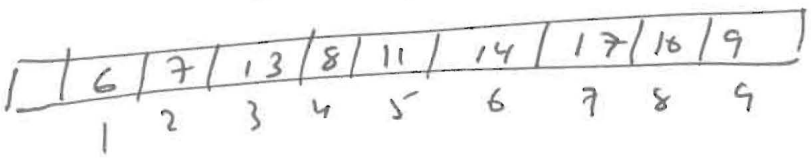
i = 3
Left = 6
Right = 7



Enqueue 8 :



i = 9
parent = i/2 = 4



i = 4
parent = 2
