1. Complete the code below for the function that increases the value of an entry in a max-heap array by the amount given, then restores the max-heap property in time ***O***(lg n), where n is the size of the heap.

Hint: “sift up”.

```
/* Note: since array is already a max-heap it can be stated that it starts with maximum values
of subtrees at the roots. When heapIncrease increases a value we only have to check the root
values up through the tree since if modified value becomes larger than parent the modified
value and parent will still be >= any other possible child after swapping places. */

void siftup (int heap[], int target);
void swapnodes (int heap[], int to_be_demoted, int to_be_promoted);

void heapIncrease (int heap[], int index, int addAmount)
{
     heap[index] += addAmount;
     siftup(heap, index);
}

void siftup (int heap[], int target)
{
     if (target != 0)
     {    /* if target == 0 it is already largest root, no furth act needed */
          int parent = (target - 1) / 2;  /* per parent = (k-1)/2 */
          if (heap[parent] < heap[target])
          {    /* parent is now smaller, reorder nodes and recursive call to check up tree */
               swapnodes (heap, parent, target);     /* swap parent and target values */
               siftup (heap, parent);     /* check up tree */
          }    /* otherwise parent is larger than child(ren) no furth act needed */
     }
}

void swapnodes(int heap[], int to_be_demoted, int to_be_promoted)
{
     int temp = heap[to_be_demoted];                  /* stores lower val */
     heap[to_be_demoted] = heap[to_be_promoted];      /* puts higher val in lower index pos */
     heap[to_be_promoted] = temp;                /* puts lower val in higher index pos */
}
```

2. You are to trace the execution of Heapsort on the following array by showing the contents of the array after each change in data positions (like swaps). The array is already a heap.

Y	H	M	D	G	K
K	H	M	D	G	Y
M	H	K	D	G	Y
G	H	K	D	M	Y
K	H	G	D	M	Y
D	H	G	K	M	Y
H	D	G	K	M	Y
G	D	H	K	M	Y
D	G	H	K	M	Y
D	G	H	K	M	Y
🟨	= sorted list contents				
⬜	= remaining heap contents				