Find K-largest elements in array

```
#include <stdio.h>

void swap(int *a, int *b)
{
   int temp = *a;
   *a = *b;
   *b = temp;
}

void minHeapify(int arr[], int size, int index)
{
   int left = 2*index+1;
   int right = 2*index+2;
   int smallest = index;
   if(left<size && arr[left]<arr[smallest])
      smallest = left;
   if(right<size && arr[right]<arr[smallest])
      smallest = right;
   if(smallest!=index)
   {
      swap(&arr[index], &arr[smallest]);
      minHeapify(arr, size, smallest);
   }
}


void buildMinHeap(int *arr, int size) {
   for(int index = size/2; index >= 0; index--)
      minHeapify(arr, size, index);
}

void printKElements(int *minHeap, int k)
{

   // print k-largest elements
   for(int index = 0; index < k; index++)
         printf("%d\t", minHeap[index]);
}

void kLargestElements(int *arr, int size, int k)
{
   buildMinHeap(arr, k);
   for(int index = k; index < size; index++)
   {
      if(arr[index] > arr[0])
      {
         arr[0] = arr[index];
         minHeapify(arr, k, 0);
      }
   }
```

```
    printKElements(arr, k);
}


int main() {
    int *arr, size, k;
    printf("Enter size of the heap");
    scanf("%d", &size);
    printf("Enter elements to heap\n");
    for(int index = 0; index< size; index++)
        scanf("%d", &arr[index]);
    printf("Enter value of k\n");
    scanf("%d", &k);
    kLargestElements(arr, size, k);
    return 0;
}
```

Time complexity: O(k + (n – k) logk) without sorted output. If sorted output then O(k + (n-k)logk + klogk)
Space complexity: O(1)