2) (10 pts) DSN (Heaps)

a) (6 pts) Consider the following struct that represents a binary minheap.

```
typedef struct heap {
    int* elements; //points to the array of heap elements
    int capacity; // total size of the array
    int size;     // actual number of elements in the heap
} heapStruct;
```

Also, the following functions are available to you, and you are free to call them as needed:

- `int removeMin(heapStruct *h);//removes the smallest item from the heap pointed to by h.`
- `int size(heapStruct* h); // returns the number of elements in the heap pointed to by h.`

Write a function called heapsort that takes a pointer to a heap, and returns those values in a sorted integer array. At the end of the function, the heap pointed to by h will be empty.

```
int* heapsort(heapStruct* h) { //complete this function

    int n = size(h);                          // 1 pt
    int* res = malloc(n*sizeof(int));         // 1 pt
    for (int i=0; i<n; i++)                   // 1 pt
        res[i] = removeMin(h);                // 2 pts
    return res;                               // 1 pt
}
```

Note: Students can access the size directly via h->size so they could use a while loop (while (h->size > 0)). Since removeMin adjusts the size of the heap, separately changing this variable in addition to the function call is incorrect.

b) (4 pts) Specify the worst run-time when efficiently implemented for the following operations:

Operation	**Run-time**
Building a binary heap from an unordered array of size **n** using heapify	O(**n**)
Inserting an item into a binary heap with **n** items.	O(**lg n**)
Deleting the minimum item from a binary heap with **n** items	O(**lg n**)
Heapsort of **n** items.	O(**nlg n**)