Lab 3: Implement a MaxHeap class to contain Integers

A. Implement a class MaxHeap (a max heap) that contains integers that represent both the priority and name of the object.

- Implement class MaxHeap of integers:
  - **def __init__(self, capacity=50)** Constructor creating an empty heap with default capacity = 50 but allows heaps of other capacities to be created.
  - **def enqueue (self, item)** . // inserts "item" into the heap, returns true if successful, false if there is no room in the heap
  - **def peek(self)** returns max without changing the heap
  - **def dequeue(self)** returns max and removes it from the heap and restores the heap property
  - **def heap_contents (self)** returns a list of contents of the heap in the order it is stored internal to the heap. (This may be useful for in testing your implementation.)
  - **def build_heap (self, alist)** Method build_heap that has a single explicit argument "list of int" and builds a heap using the **<u>bottom up method</u>** discussed in class. It should return True if the build was successful and False if the capacity of the MaxHeap object is not large enough to hold the "array of int" argument.
  - **def is_empty(self)** returns True if the heap is empty, false otherwise
  - **def is_full(self)** returns True if the heap is full, false otherwise
  - **def capacity(self)** --- this is the maximum number of a entries the heap can hold. Note that this is one less than the number of entries that the array allocated to hold the heap can hold.
  - **def size(self)** -- the actual number of elements in the heap, not the capacity
- Where possible your build, insert, and delete methods should use the following functions that work exactly as described in class. **Normally these would be private but make them public for testing purposes.**
  - **def perc_down(self,i)** where the parameter **i** is an index in the heap and perc_down moves the element stored at that location to its proper place in the heap rearranging elements as it goes. Since this is an internal method we will assume that the element is either in the correct position or the correct position is below the current position.
  - **def perc_up(self,i):**. similar specification as perc_down, see class notes

B. Add a function heap_sort_a (alist) to perform heap sort given a list of positive integers.

Add a function to the MaxHeap Class to sort a list of positive integers in increasing order.

- **heap_sort_ascending (self, alist)** takes a list of integers and returns a list containing the integers in non-decreasing order using the <u>Heap Sort algorithm as described in class</u>. Your MaxHeap class must use the list internal to the heap to store the sorted elements, as described in class. This will result in the list internal to the heap being sorted in increasing order. This enables the reuse of the space but will destroy the heap order property. However, then you can just return the appropriate part of the internal list since you will not be using the heap anymore.

Submit a file **heap_lab.py** containing the above and a file **heap_file_tests.py** containing your set of test cases. Your test cases should test all the functions. Some are quite simple to test but some will require multiple test cases. Again, developing test cases as you develop each function will save you time overall.