Project 2: Implementation of the Binary Heap ADT and Its Application

Part I. Design and implement a Binary Heap ADT.

1) Define a class called *BinHeap* to represent a binary heap of *any* type objects.
BinHeap has a nested class *MyException.*
BinHeap has two instance variables – an array to hold the collection, and an integer to hold the actual number of elements in the heap (the size of the heap).
BinHeap has an *__init__()* method that creates an empty heap**:**
- *__init__()*: has one *int* type parameter for the size of the array. It allocates memory for the array that will hold elements of the heap (the length of the array is defined by the parameter value, see initialization of QueueArray for reference) and assigns the size of the heap to be 0.

BinHeap has **five** methods (you may include support methods for them):
- *insert*: *element* parameter; no return value. The parameter element is added to the heap using the corresponding algorithm, and the size of the heap is incremented.

`Important:` if the array is full, you need to resize it. So, **BEFORE** you add a new element, you need to check if the array is full or not, and if it is, you need to create a new array **twice the size** of the original array and copy the heap content from the old array to the new one (all elements in the old array are copied into the matching indexes of the new array). Once the new array is "activated" (taken as your main array), you can then go ahead with adding the new item as usual.
- *deleteMin*: no parameters; returns and element value – the smallest element in the collection. The element at the root of the heap is removed using the corresponding algorithm, and the size of the heap is decremented. The removed element is returned by the method.

A *MyException* type exception is raised **if the heap is empty** (check the size of the heap).

- *isEmpty*: no parameters; returns *true* if the heap has no elements, and *false* otherwise.

- *size*: no parameters; returns an *int* type value – the size of the heap (the number of elements **in the collection**, NOT in the array).

- *__string__* or *__repr__*: no parameters; returns a *String* type value – a string containing all **elements of the heap** (i.e. content of array cells containing heap elements) **separated by at least one space**, in the **order they are stored** in the array. Attention: do not output the whole array.

Note: this method is very useful for testing purposes.