H2 Computing Practical Worksheet – T1W7

1a Write the object oriented code for a Binary Search Tree.

Your implementation should implement a BST that is initialised with a specified maximum size, and that stores its objects statically within an array. More specifically, you are to adopt a Heap Implementation. For more details, refer to:
https://en.wikipedia.org/wiki/Binary_heap#Heap_implementation

The objects stored in this BST should still be generic – i.e., it should be able to store any type of data object. It will thus be the responsibility of the user to overload any comparison functions for that data object so that the BST will function properly.

Your BST must include the following methods:

- Initialisation – With max size of the BST as input.
- Insertion – Maintaining BST ordering.
- Find – Returns True if the data object input can be found in the BST, or else returns False.
- Delete – Given a data object input, finds and deletes the object in question. Returns True is successful, or else returns False. Note that you may implement this using Tomb-stoning. However, if you do so, your find and insertion methods must account for this.
- Print – Outputs the contents of the heap – i.e., the array; this should call the print method of the objects stored, and print "empty" for cells in the array that are empty.
- Print Pre-order – Prints the contents of the BST based on an pre-order traversal.
- Print In-order – Prints the contents of the BST based on an in-order traversal.
- Print Post-order – Prints the contents of the BST based on an post-order traversal.

Your BST class should also adhere to the standard object oriented design principles, including encapsulation and data hiding. All methods should be modular and utilise helper methods if applicable.

You are also expected to adhere to good UI principles when designing any output.

1b Design and implement a series of tests that evaluate the functionality of your BST implementation. Your test should be conducted over the following data objects (separately – i.e., each test should only utilise 1 type of data object):

- integers
- strings
- a Student data object consisting of 2 attributes: Name and NRIC (not that your tests should include using both attributes as the attribute to sort over – this will require comparison overloading for the Student object you implement).

Your design should be included as comments accompanying your test script.