[6 MARKS]

We have provided you with a file called **Q2_answers.txt**. Use PyCharm to open that file and add your answers to it. Hand in this file on MarkUs.

Here is the documentation for the binary search tree class that you worked on many times this term:

```
class BinarySearchTree:
    """This class represents a binary tree satisfying the Binary Search Tree
    property: for every item, its value is >= all items stored in its left
    subtree, and <= all items stored in its right subtree.
    """
    # === Private Attributes ===
    _root: Optional[Any]
    _left: Optional[BinarySearchTree]
    _right: Optional[BinarySearchTree]

    # === Representation Invariants ===
    #  - If self._root is None, then so are self._left and self._right.
    #    This represents an empty BST.
    #  - If self._root is not None, then self._left and self._right
    #    are BinarySearchTrees.
    #  - (BST Property) If self is not empty, then
    #         all items in self._left are <= self._root, and
    #         all items in self._right are >= self._root.
```

Below are three methods we added to this class. For each one, name the method and answer these questions:

1. Does the method ensure that the BST property holds after it has been called? (Assume that it was called on an instance of the class that satisfies the Representation Invariants.)

2. If it does not, give (a) an example tree for which a call to the method will cause a violation of the BST property (label this "input"), and (b) the tree as it will be after the method returns (label this "output"). See the next page for how to "draw" your trees in a text file.

```
def carrot(self) -> None:
    if not self.is_empty() and not self._right.is_empty():
        self._root = self._right._root

def eggplant(self) -> None:
    if not self.is_empty():
        if not self._left.is_empty() and not self._right.is_empty():
            self._root = self._left._root + self._right._root
            self._left.eggplant()
            self._right.eggplant()

def peanut(self) -> None:
    if not self.is_empty():
        self._root = 1
        self._left.peanut()
        self._right.peanut()
```

"Draw" your trees using an indented format, where each subtree is followed by its left and right subtree, each indented. If one of the subtrees is missing, put a dash (-). For example, the text on the left represents the tree on the right:

```
7
    3
        2
        5
    11
        -
        13
```