5 A binary search tree holds the names of capital cities in the array `bst`. Each element of the array contains three values. `leftPtr` and `rightPtr` are integers and `dataValue` is a string.

leftPtr	dataValue	rightPtr

The root of the binary search tree is stored in an integer variable, `rootPtr`.

The unused elements of the array are in a free space list that starts from `freePtr`.

The provided names of capital cities are guaranteed to be unique.

The contents of the array `bst` is shown below. –1 represents the null pointer.

Index	leftPtr	dataValue	rightPtr
0	5	London	1
1	2	Paris	3
2	–1	New York	–1
3	–1	Rome	4
4	–1	Tokyo	–1
5	–1	Beijing	–1
6	7		–1
7	8		–1
8	9		–1
9	–1		–1

rootPtr
0

freePtr
6

An individual value in the array can be accessed in the following format:
`bst[0].leftPtr`

(a) **(i)** Draw the binary search tree represented by the array `bst`. [2]

(ii) Give **two** properties of a binary search tree. [2]

(iii) The city of Madrid is to be inserted into the binary search tree.
Identify the changes that will be made to any pointers and the contents of the array `bst`. [3]

(b) `bst` has been implemented as a static data structure of 10 nodes.

State **two** problems that can arise from using a static data structure. [2]

(c) A reverse in-order traversal of a binary tree can be written recursively as:

1. follow the right pointer and recursively repeat from step 1
2. output the current node
3. follow the left pointer and recursively repeat from step 1

(i) Write a recursive pseudo-code procedure `reverseInOrder()` that takes the value of `rootPtr` and outputs the cities in `bst` using a reverse in-order traversal. [6]

(ii) Give the output of performing the reverse in-order traversal on `bst` from part **(a)**. [1]