# Exercise 2

This exercise has total 6 points. Each point amounts to one percent of the total module mark. You are not allowed to use any external library in your code, doing so will receive a mark of 0.

*All assessment deadlines in this module are strict. Late submissions will get a mark of 0. All submissions must work on the Linux lab machines as specified.*

A good way to understand the challenges of pointers and memory management is to implement tree data structures. A binary search tree will be implemented using the structure struct Node.

```
struct Node{
   int data;
   struct Node *left;
   struct Node *right;
};
typedef struct Node Node;
```

See tree.c file.
Implement the following functions for performing operations on binary search trees.

1.

Node* insertNode(Node *root, int value);

This function is used to insert a new node in the tree as a leaf node. The member 'data' of the new node is assigned the input 'value'. During the insertion, a traversal starts from the root of the tree. The traversal follows the right branch if the value of the new node is greater than the value of the currently visited node. Otherwise the traversal follows the left branch.

2.

Node* deleteNode(Node *root, int value);

This function is used to delete an existing node in the tree. For this question we will restrict there to being unique data values (that is to say there will be no duplicates) for simplicity.

3.

void printSubtree(Node *N);

This function is for Inorder printing of the node values. It prints i) the values in the left subtree ii) then the value of the node and iii) finally the values in the right subtree. The function prints the node values in sorted ascending format.

4.

int countNodes(Node *N);

This function returns the number of nodes in the subtree rooted at node N.

5.

void freeSubtree(Node *N);

This function deallocates the entire subtree rooted at N. Here you will need to free each node exactly once, so that there are neither double frees nor memory leaks. Note that the tree could be empty, so freeSubtree() should work, but do nothing, as there are no nodes to free.

## Code submission

Compile the code: gcc -std=c99 -Werror -Wall tree.c -o tree

Use Valgrind to check memory leaks and errors.

Upload the tree.c file **ONLY**, any other form of submission will not be accepted.

**Marking scheme:** We will test your program with arbitrary inputs.

- Your code must compile on the lab machines as above. If it fails to compile or produces any errors (including memory errors), it will be given 0 marks.
- 1 point is given if insertNode() inserts correctly.
- 2 points are given if deleteNode() deletes correctly.
- 0.5 points are given if printSubtree() prints the subtree correctly (i.e in ascending order)
- 0.5 points are given if countNodes() returns the number of nodes correctly
- 1 point is given if freeSubtree() deallocates the subtree without causing memory leaks
- 1 point is reserved for exceptional quality code (e.g. readablility, corner cases, novelty)

We remind you that plagiarism of submissions will be detected and handled in accordance with the University's policies.