MM560 – Introduction to Programming – 2023

Lecturer:	Benjamin Jäger	E-Mail: jaeger@imada.sdu.dk
Tutor:	Sofie Martins	E-Mail: martinss@imada.sdu.dk

Exam Project 2 - Python

Submission deadline 23.59 on the 08.06.2023

Submission

As announced before, we allow for project submission in teams of (at most) two. The idea is to exchange ideas and solutions to problems rather than codes. You can work on the code together, but both should work on both projects. However, each team member has to submit the project code individually through ItsLearning. Include your name and the name of your team partner (if applicable) in the header of every code file that you submit.

For example:

```
% Name: <Your Name>
% Team partner: <Your team partners name>
%
% (Additional comments...)
% code starts here
```

Submit the code to ItsLearning (if it is more than one file, please compress it into a zip file). If you have any problems, contact me (before the deadline).

Code without comments will be considered nonexistent, so please add comments to **all** parts of the code.

1) Binary Search Tree in Python [50 Points]

A Binary Search Tree (BST) is a data structure that stores data in a tree-like structure. The root is the top of the tree. Every element in the tree contains a left and a right Node. Smaller numbers are stored in the left sub-tree, larger in the right sub-tree. The **Node** class realises the BST in Python. An example BST is shown in Figure 3. The corresponding UML diagram is shown in Figure 4. A Binary Search Tree consists of a root, a left and a right sub-tree.

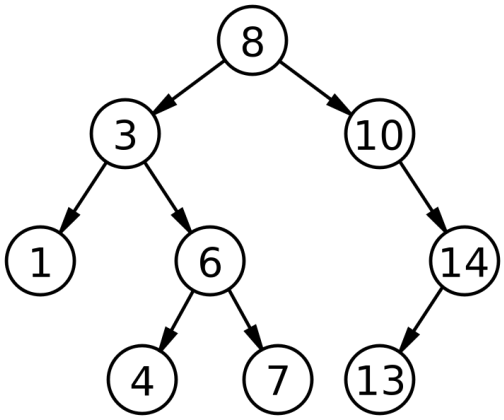

Figure 1: Example Binary Search Tree

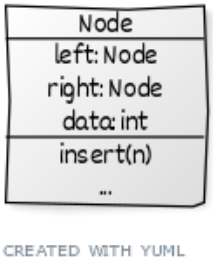

Figure 2: UML diagram of BST

(a) Load the script and the BST template from ItsLearning. Run the script and make sure it works. Try it with a different set of numbers. [4 Points]

(b) Study the provided **insert(val)** function and add comments to the method to explain its functionality. [4 Points]

(c) Write a **print()** function that prints out the whole tree. You can choose how you want to print out the tree. At least every element shall be listed.
(Hint: You can use a recursive function that calls the print function from the left and the right nodes). [6 Points]

(d) Write a **heigth()** function that computes the height of the tree. The height is defined as the distance between the root and the lowest leaf. For example, the height of tree in Figure 3 is 3. [6 Points]

(e) Write a **minumum()** and **maximum()** function that obtains the smallest entry of the BST. [6 Points]

(f) Write a **numberLeaves()** function that obtains the number of leaves (the root counts as one leaf). For example, the tree in Figure 3 has 9 leaves. [6 Points]

(g) Write a **find(val)** function that finds the node that contains val. If no such node exists return an emtpy list. [6 Points]

(h) Write a **sum()** function that computes the total sum of all nodes in the BST. [6 Points]

(i) Write a **sumGreaterThan(val)** function that computes the sum of all elements greater than val. [6 Points]