CS 234 Fall 2019
Naomi Nishimura

Assignment 3

Coverage: Modules 4-6.

This assignment consists of a written component and a programming component. Please read the course website carefully to ensure that you submit each component correctly. For full marks, you are required not only to have a correct solution, but also to adhere to the requirements of the assignment question and the style guide, including aspects of the design recipe.

All the questions in the written component should be submitted as a single pdf file with the name `a3written.pdf`. Although scanned documents are acceptable, please keep in mind that only legible writing will be marked (subject to the markers' discretion) and that MarkUs forces a limit of 5000 KB per file, which might be exceeded by high resolution scans.

Several of the questions will make use of the ADT Binary Tree, as discussed in class. In the list of operations below, B is a binary tree and *Node* is a node in B. The list is not intended to be comprehensive, but should be sufficient for you to complete the exercises.

Name	Returns
CREATE*()*	a new empty binary tree
IS_EMPTY*(B)*	*True* if empty, else *False*
ROOT*(B)*	the root node of B, if any, else *False*
VALUE*(B, Node)*	value of key at *Node*
PARENT*(B, Node)*	the parent of *Node*, if any, else *False*
LEFT_CHILD*(B, Node)*	the left child of *Node*, if any, else *False*
RIGHT_CHILD*(B, Node)*	the right child of *Node*, if any, else *False*

Written component

For full marks, you are expected to provide a brief justification of any answer you provide.

W1. *[7 marks]* In this question you will consider how to determine whether a node in a binary tree is a leaf, either using operations of the ADT Binary Tree or by augmenting the ADT to include a new operation.

(a) *[2 marks]* Using the ADT Binary Tree, write pseudocode for the function IS_LEAF that consumes a tree and a node *Node* and produces *True* if *Node* is a leaf and *False* otherwise.

(b) *[2 marks]* Suppose that we wished to augment the ADT Binary Tree with the operation IS_LEAF. Explain how the algorithm would be implemented when the binary tree is implemented using linked nodes, and give and briefly justify the running time of your implementation.

(c) *[3 marks]* Suppose that we wished to augment the ADT Binary Tree with the operation IS_LEAF. Explain how the algorithm would be implemented when the binary tree is implemented using a contiguous implementation, and give and briefly justify the running time of your implementation.

W2. *[5 marks]* Using the ADT Binary Tree, write pseudocode for the function FIRST that consumes a tree and a node *Node* and produces the node in the subtree rooted at *Node* that will appear first in an inorder ordering of the nodes in the subtree. For example, for the tree in Figure 1, in the subtree rooted at 2, the node 8 will be the first in inorder ordering, and in the subtree rooted at 6, the node 6 will be the first in inorder ordering. For full marks, give a brief explanation of the rationale behind your pseudocode; that is, explain why what is produced will appear first in an inorder ordering. Your solution should not use a function that generates an inorder traversal of the tree.

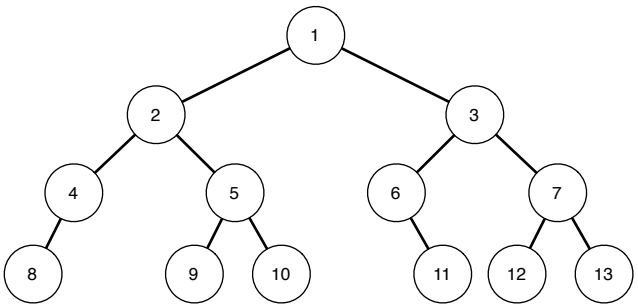

Figure 1: A binary tree

W3. *[8 marks]* Describe an algorithm SUCCESSOR that consumes a binary tree and a node *Node* in the tree and produces the node that is the inorder successor of *Node* (that is, the node that appears right after *Node* in the inorder ordering), if any, and otherwise *False*. You may wish to use one or more function applications of FIRST. You do not need to have correctly completed the previous question to receive full marks on this question. Your solution should not use a function that generates an inorder traversal of the tree.

Pseudocode is not required for this question. You may find it useful to organize your algorithm by breaking the explanation into cases, explaining what the inorder successor is in each case and how the algorithm will find it.

Programming component

Please read the course website carefully to ensure that you are using the correct version of Python and the correct style. For full marks, you are required not only to have a correct solution, but also to adhere to the requirements of the assignment question and the style guide, including aspects of the design recipe.

Although submitting tests is not required, it is highly recommended that you test your code. For each assignment question, create a testing file that imports your submission and tests the code. Do not submit your testing file.

You may import any of the following files: `check.py` and `boundeddegreenode.py`.

P1. *[20 marks]* In this question you will implement the ADT Undirected Graph for the special case in which there is a known bound b on the degree of each vertex. Each vertex will be implemented as a linked node that contains various attributes of the vertex (ID, value,

weight, and colour) and links to each of its at most b neighbours. To define the graph in as generic a way as possible, instead of having separate functions for checking (or setting) value, label, and colour, we instead use a method `check_vertex_attribute` that consumes both a vertex and a name of an attribute, as well as a method `set_vertex_attribute` that consumes a vertex, a name of an attribute, and the new value to be set.

The nodes have been implemented for you in the file `boundeddegreenode.py` and the interface for the operations has been provided for you in the file `boundeddegreegraphinterface.py`, both linked off the course web site.

You might find it convenient to use the module `equiv.py` for testing; both a description of the module and the module can be accessed from the assignments page of the course website. b Submit your work in a file with the name `boundeddegreegraph.py`.