UNIVERSITY OF BIRMINGHAM

School of Computer Science

Data Structures & Algorithms Mock Exam

Mock Exam

Data Structures & Algorithms Mock Exam

Note

Answer ALL questions. Each question will be marked out of 20. The paper will be marked out of 60, which will be rescaled to a mark out of 100.

1. Stacks are often used to implement *"undo"* operations in applications like text editors or Web browsers such that making a change in the text or clicking on a link in the browser pushes a record of that operation on to the stack, while invoking undo pops the record off the stack and executes the actions necessary to undo the operation.

   In theory, an unbounded stack can provide unlimited undo operations, but, to save memory, many applications support only limited undo history. Thus invoking *"push"*, when the stack is already at full capacity, accepts the pushed elements at the top of the stack and *leaks* the oldest element from the bottom of the stack rather than throwing an exception, while invoking `pop` when the stack is empty still throws a `StackEmptyException`.

   (a) Describe, as clearly and simply as possible, your proposed strategy to implement such a stack using a fixed-capacity array, so that each update operation runs in $O(1)$ time. **[6 marks]**

   (b) Write the pseudocode for the `push` operation of such a stack. **[7 marks]**

   (c) Write the pseudocode for the `pop` operation of such a stack **[7 marks]**

2. An inefficient recursive function `isbst` was presented in the module in the handout document (pages 46–47) and the slides (dsa-slides-04-03-binary-search-trees.pdf).

   Write an efficient, recursive `isbst` function in pseudocode to check that a binary tree is a valid Binary Search Tree based on checking that subtrees are within closed intervals starting with [MIN_INT, MAX_INT] and calculate the complexity, in terms of $O(g(n))$, with respect to the size of the tree. Justify your calculation of the complexity with a short explanation. **[20 marks]**

3. Construct an AVL tree of positive integers and of height 3 such that insertion of the value 7 will trigger a double rotation (i.e. a right rotation followed by a left rotation or a left rotation followed by a right rotation) and, following the insertion of 7, an insertion of the value 8 will also trigger a double rotation.

   (a) Draw the valid AVL tree before the specified insertions. **[6 marks]**

   (b) Draw the valid AVL tree after insertion of 7 but before insertion of 8. **[7 marks]**

   (c) Draw the valid AVL tree after insertion of 7 and of 8 in that order **[7 marks]**