University of Nottingham

SCHOOL OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

A LEVEL 2 MODULE, AUTUMN SEMESTER 1999–2000

ALGORITHMS AND DATA STRUCTURES
(Course G52ADS)

Time allowed TWO hours

Candidates must NOT start writing their answers until told to do so

Candidates should attempt FOUR out of six questions

Marks available for sections of questions are shown in brackets in the right-hand margin

1 (a) Describe how insertion sort works. Give pseudocode (or Java code) for insertion sort. (5)

(b) What is the worst-case time complexity of insertion sort? Give a sketch of a proof. (5)

(c) Suppose that you are writing an application using a library where sorting is implemented using insertion sort. Suppose further that your application uses sorting a lot and you do not want it to spend more than 2000 milliseconds sorting a typical input. Assume that the library sort method sorts an array of length 10 in 5 milliseconds and an array of length 100 in 495 milliseconds. For which size of typical input (approximately) would you consider writing your own sorting method, and which algorithm would it implement? Explain why. (10)

(d) Would your answer to question 1(c) remain the same if your application sorted lists instead of arrays? Explain your answer. (5)

2 A maze is a collections of rooms connected by corridors. It may have several exits (assume that exits are a special kind of room). One can only get from one room to another by following a corridor. Suppose that you are writing a game which involves implementing a maze, navigating from one room to another and finding an exit.

(a) Which data structure would you use to store data about the maze? (2)

(b) How would you implement that data structure? Illustrate your answer using the following maze:

```
                        _________        _________
__________             | room 3 |       | room 5 |
|         |____________|        |_______|        |____________
| room 1  ______ _________  __________________________       |
|_________|    | |       | |        _________________| room 7|
               | |       | |________|           _____________|
 _________     | |       |  ________  room 6    |
 room 2   |____| |       | |        |___________|
 (exit)   _______|      _| |_
 _________|            |      |
                       |room 4|
                       |(exit)|
```

(5)

(c) Which algorithm would you use to implement a method `escape(room)` which finds a path to the exit, and why? Would it find the shortest path (with the least number of rooms on it)? (3)

(d) Explain how the algorithm which you have chosen in question 2(c) works (in English or in pseudocode). (10)

(e) Trace the algorithm on the example maze from question 2(b) for room 7. (5)

3 (a) What is a complete binary search tree? Draw a complete binary search tree with 6 nodes storing numbers $40, 50, 60, 70, 80, 81$. (5)

(b) Describe two possible implementations for complete binary trees. Show how your tree from question 3(a) would be implemented in each of them. (5)

(c) Describe the search algorithm for binary search trees. For one of the implementations of complete binary trees, give pseudocode for the search method. Trace the search for 60 on your tree from question 3(a). (10)

(d) How many comparisons in the worst case would the search algorithm make searching for an item in a complete binary tree holding 500 nodes? (5)

4 (a) Design a linear time algorithm for sorting an array of positive integers which relies on the fact that all integers in the array are less than some integer k. Assume that all numbers in the array are different. (10)

(b) Do you need to change your algorithm if the array may contain repetitions? If yes, describe the changes. (5)

(c) Prove that the worst-case complexity of your algorithm is O(N). (5)

(d) What is the space complexity of your algorithm? (5)

5 Given is an array c, indexed from 0 up to (and including) $N-1$, such that each value in the array is either red or blue. It is required to sort the elements of the array so that, for some r, the first r values in the array are all red and the remaining $N-r$ values are blue.

(a) Construct an algorithm to sort the array that uses two indices r and b and maintains the invariant property

$$(0 \le r \le b \le N) \ \wedge\ \forall\langle i : 0 \le i < r : c[i] = \mathit{red}\rangle \ \wedge\ \forall\langle i : b \le i < N : c[i] = \mathit{blue}\rangle$$

(10)

(b) What is the bound function for your algorithm? (5)

(c) Construct verification conditions for the initialisation, loop body and termination of your algorithm which demonstrate the algorithm's correctness. (10)

6 A simple spreadsheet allows the user to create new rows and columns, insert and delete data in cells (intersections of rows and columns), sum up numerical data in a given row or column and sort data in the table in accordance with order in some column. For simplicity, assume that all data is numerical (of type `double`).

A Spreadsheet ADT has the following methods:

- void initialize()
  Postcondition: a new empty spreadsheet is created.
- void addRow()
  Postcondition: a row is added to the spreadsheet.
- void addColumn()
  Postcondition: a column is added to a spreadsheet.
- void insert(row, column, item)
  Precondition: the row and the column exist.
  Postcondition: the item is inserted at the intersection of the row and the column.
- void delete(row, column)
  Precondition: the row and the column exist.
  Postcondition: deletes the item at the intersection of the row and the column.
- double sumRow(row)
  Precondition: the row exists.
  Postcondition: the sum of the numbers in the row is returned
- double sumColumn(column)
  Precondition: the column exists.
  Postcondition: the sum of the numbers in the column is returned
- void sort(column)
  Precondition: the column exists.
  Postcondition: sorts the rows in the table in accordance with the order in the specified column.

Describe the data structures and algorithms you would employ to implement the Spreadsheet ADT, explaining how they are to be used, and the reasons for your choices. Give pseudocode for the methods. (25)