

Module Code: 7SENG010W

Module Title: Data Structures & Algorithms

Date:

Start Time: (TIME 90 minutes)

Time Allowed: 1 Hour, 30 minutes

RAF Time Allowed: 2 Hours

INSTRUCTIONS FOR CANDIDATES

ICT ANSWERS

Answer ALL questions.

University of Westminster

School of Computer Science & Engineering

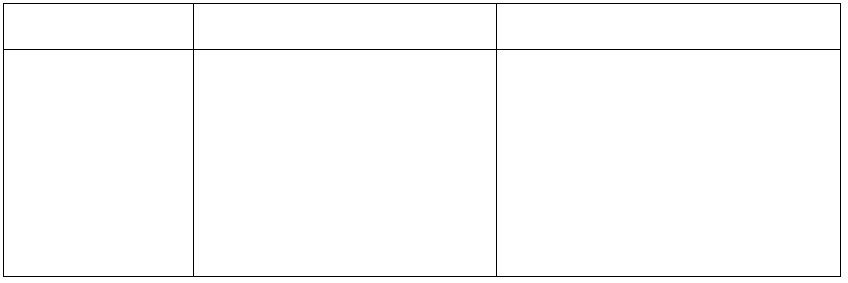
Module Code: 7SENG010W

Module Title: Data Structures & Algorithms [MARKING SCHEME]

Question 1

(a) The input size N doubles between inputs, and the run-times T(N) grow

by a factor of approximately 2:

 Input Size Milliseconds (ms) T(2N)/T(N) ≈ 2

1000 473 –

2000 1040 1040 / 473 = 2.2 4000 1934 1934 / 1040 = 1.86 8000 4003 4003 / 1934 = 2.0716000 7765 7765 / 4003 = 1.94

So T(2N)/T(N) ≈ 2. [2 marks]Since 2 = 21 (or log2(2) = 1) the algorithm is linear, T(N) = N. [1 mark]

Therefore, the algorithm’s Big-O complexity class is O(N1) = O(N). [2 marks]

[ PART Total 5]

(b) The outer while-loop i iterates n times. The middle for-loop j iterates

up to i which iterates to n, so j iterates up to n. The inner while-loop k iterates up to j which iterates up to n, so k iterates up to n. [3 marks]

So total iterations are n ∗ n ∗ n = n3, hence T(n) = n3. [1 mark]

Therefore the Big-O order of complexity for the Fun1 method is cubic, i.e. O(n3). [1 mark]

[ PART Total 5]

[QUESTION Total 10]

Question 2 Search steps & comparisons of the BinarySearch method on the array

index 0 1 2 3 4 5 6 7 8 9

Numbers1 9 14 19 23 28 34 40 47 55 75

ꢀc University of Westminster Page 2

School of Computer Science & Engineering

Module Code: 7SENG010W

Module Title: Data Structures & Algorithms [MARKING SCHEME]

(a) BinarySearch( Numbers1, 23 ) steps:

1) Numbers[4] = 28: array middle, start=0, middle=4, end=9, & 23 < 28,2) Numbers[1] = 14: middle of left segment, s=0, m=1 e=3, & 14 < 233) Numbers[2] = 19: middle of left segment, s=2, m=2, e=3, & 19 < 234) Numbers[3] = 23: middle of right segment, s=3, m=3, e=3, & 23 = 235) Found 23 at index 3, so returns index 3.

[5 marks]

(b) BinarySearch( Numbers1, 63 ) Steps:

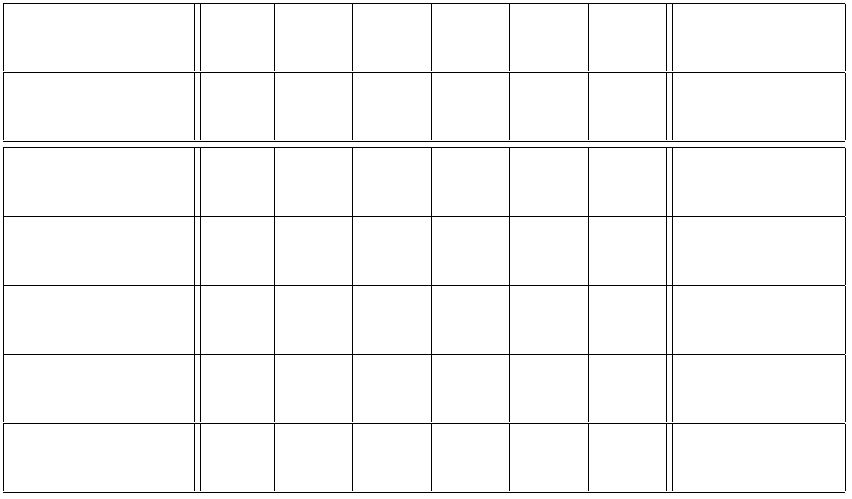
1) Numbers[4] = 28: middle of array, s=0, m=4, e=9, & 28 < 632) Numbers[7] = 47: middle of right segment, s=5, m=7, e=9, & 47 < 633) Numbers[8] = 55: middle of right segment, s=8, m=8, e=9, & 55 < 634) Numbers[9] = 75: middle of right segment, s=9, m=9, e=9, & 63 < 755) Search left segment s=9, e=8, but its empty, so 63 not found, return-1.

[5 marks]

[ PART Total 10]

Question 3

Selection Sort algorithm on the Numbers2 array, showing updated state of array& the two numbers swapped during the pass. The 2 options are sort highestnumbers at the end of the array or the lowest numbers at the start of the array,either is acceptible. Sort from the end:

Index 0 1 2 3 4 5

Numbers2 53 42 79 65 33 24 Swap

(1) 53 42 24 65 33 79 24 ↔ 79(2) 53 42 24 33 65 79 33 ↔ 65(3) 33 42 24 53 65 79 33 ↔ 53(4) 33 24 42 53 65 79 24 ↔ 42(5) 24 33 42 53 65 79 24 ↔ 33

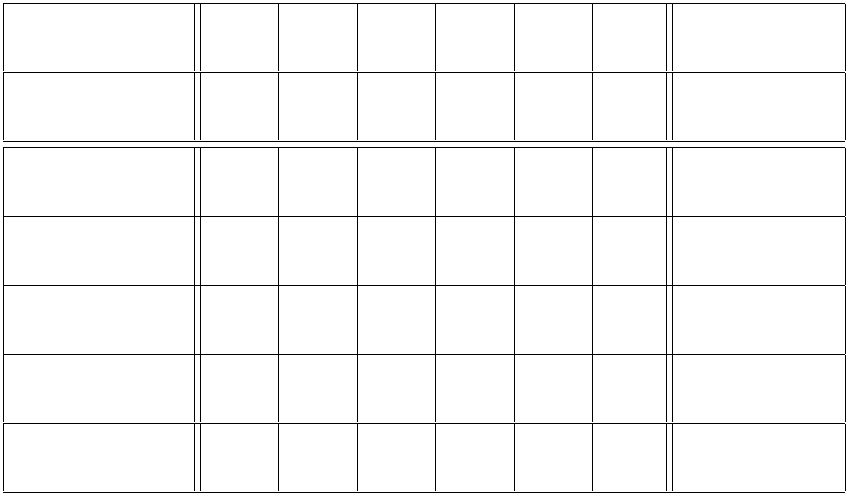
ꢀc University of Westminster Page 3

School of Computer Science & Engineering

Module Code: 7SENG010W

Module Title: Data Structures & Algorithms [MARKING SCHEME]

Sort from the start:

Index 0 1 2 3 4 5

Numbers2 53 42 79 65 33 24 Swap

(1) 24 42 79 65 33 53 24 ↔ 53(2) 24 33 79 65 42 53 33 ↔ 42(3) 24 33 42 65 79 53 42 ↔ 79(4) 24 33 42 53 79 65 53 ↔ 65(5) 24 33 42 53 65 79 65 ↔ 79

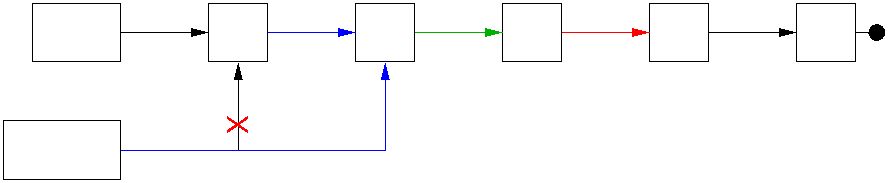
[10 marks]

[ PART Total 10]

Question 4

Step 1: search sllist for the previous node to 99 which is 10, since previous.next.item == 99

and return its link b:

head a 42 b 10 c d

99 23 60

e

a

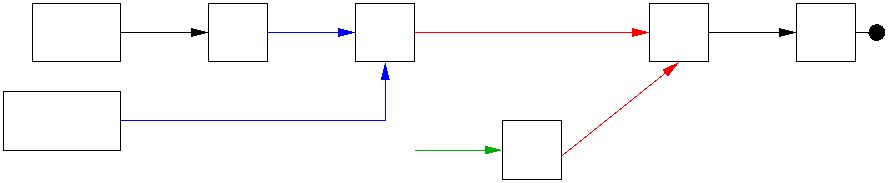
b

previous

[4 marks]

Step 2: disconnect node 99 from sllist by setting previous.next (99) to previ-ous.next.next (23), i.e. replace 10s link c with link d:

d head a 42 b 10

e

23 60

b

previous d

c

99

[4 marks]

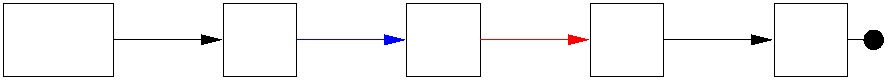
Step 3: Completed deletion of 99 from sllist, with resultant sllist:

ꢀc University of Westminster, Page 4

School of Computer Science & Engineering

Module Code: 7SENG010W

Module Title: Data Structures & Algorithms [MARKING SCHEME]

a b d e

head 42 10 23 60

[2 marks] [QUESTION Total 10]

ꢀc University of Westminster Page 5

School of Computer Science & Engineering

Module Code: 7SENG010W

Module Title: Data Structures & Algorithms [MARKING SCHEME]

Question 5

See ICT paper for the binary tree.

(a) The height of the sub-tree with root 12 is 3.

The height of a tree is the number of nodes on the longest path from theroot to a leaf, e.g. the sub-tree’s root node is 12 to the leaf node 31 is 3nodes. [2 marks]

The node values that are at level 2 in the tree are: 82, 16, 9, 97.

The level of a node is the “distance” from the node to the root of the treein terms of the number of branches, e.g. there are 2 branches from theroot node 51 to nodes 82, 16, 9, 97. [2 marks]

(b) Tree’s nodes in:

In-Order traversal: 31, 82, 12, 16, 51, 9, 64, 43, 97, 25.

[3 marks]

Post-Order traversal: 31, 82, 16, 12, 9, 43, 25, 97, 64, 51 [3 marks]

[QUESTION Total 10]

Question 6

(a) BST: a binary tree (i.e. empty or having two sub-trees) such that the value

of the key stored at any node is greater than all values in that node’s leftsub-tree, and less than or equal to all values in that node’s right sub-tree.

(Alternative: is left sub-tree is also “equal to” & right sub-tree is onlygreater.)

[2 marks]

(b) The 8 BSTs constructed in turn from the sequence of numbers:

50, 30, 61, 48, 17, 72, 66, 91

50 50 50 50 / / \ / \

30 30 61 30 61 ...

\

48

ꢀc University of Westminster Page 6

School of Computer Science & Engineering

Module Code: 7SENG010W

Module Title: Data Structures & Algorithms [MARKING SCHEME]

50

/ \

30 61

/ \ \

17 48 72

/ \

66 91

BSTs for: 50, 30, 61. [1 mark] ,

BSTs for: 48, 17, 72, 66, 91. [5 marks]

(c) Final BST from part (b) after 50 has been deleted, there are two valid

options to replace 50 with either(a) the max value in left sub-tree – 48 or(b) min value in right sub-tree – 61then reorganise BST accordingly.

(a) 48 / \

30 61/ \

17 72

/ \

66 91

OR

(b) 61

/ \

30 72

/ \ / \

17 48 66 91

[2 marks]

[QUESTION Total 10]

Question 7

(a) It is not a valid AVL tree because it is not balanced. The node 35 is

unbalanced, because it is has a balance factor of 2.

ꢀc University of Westminster

School of Computer Science & Engineering

Module Code: 7SENG010W

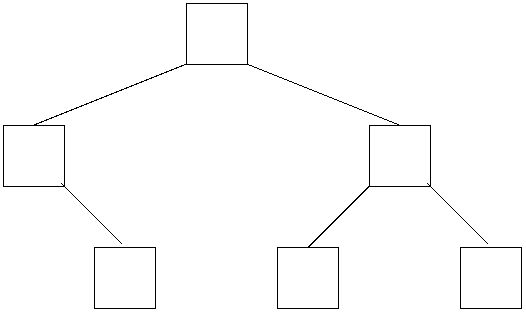
Module Title: Data Structures & Algorithms [MARKING SCHEME]

The BF of a node is the diﬀerence between the heights of a node’s leftand right sub-trees, so 35 is unbalanced because:

BF(35) = height(leftSubtree) − height(rightSubtree) = 2 − 0 = 2.So BF(35) = 2 indicates that the tree is unbalanced, as balanced nodeshave BFs of -1, 0, 1. [4 marks]

(b) The re-balanced AVL tree (see Figure 1) is produced by applying a Right-

rotation on node 35.

25

10 33

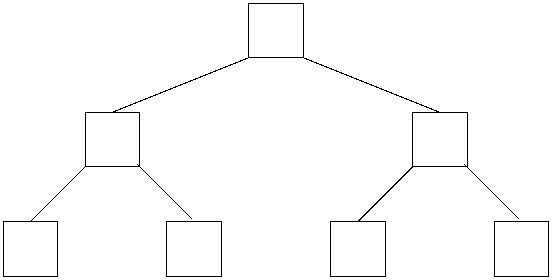
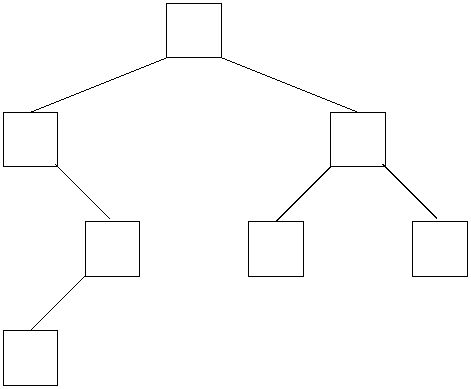
20 28 35

Figure 1: AVL tree after 28 inserted

[6 marks]

(c) See Figure 2 (a) for the initial unbalanced tree after 12 is added, with

BF(10) = -2. The re-balanced AV L tree with 12 added is given in (b). Itis produced by ﬁrst applying a Right-rotation about node 20, followed bya Left-rotation about node 10.

25 25

10 33 12 33

20 28 35 10 20 28 35

12

(a) Tree after inserting 12 (b) Re−balanced AVL tree

Figure 2: (a) Unbalanced tree after 12 inserted, (b) Re-balanced AVL tree

[10 marks]

[QUESTION Total 10]

ꢀc University of

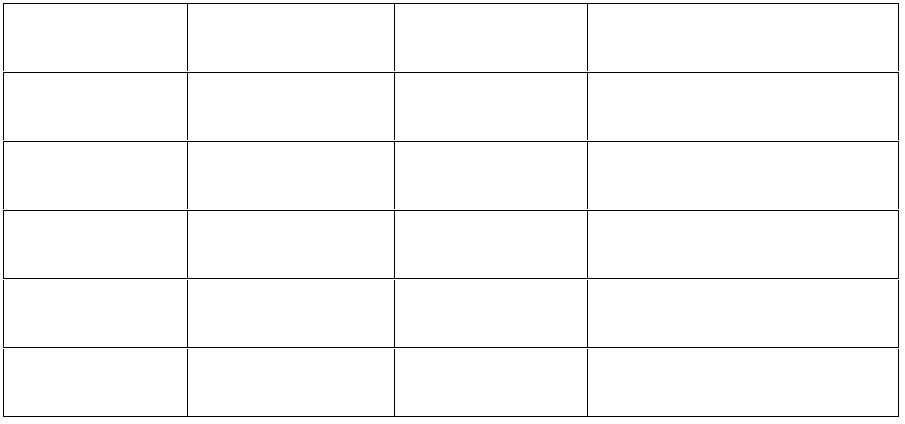
School of Computer Science & Engineering

Module Code: 7SENG010W

Module Title: Data Structures & Algorithms

Question 8

Result of applying Dijkstra’s Single Source Shortest Path algorithm to the graphwith source A:

Vertex V edgeTo V distTo V Path A to V

A - 0 A

B (A, B, 4) 4 A, B

C (A, C, 9) 9 A, C

D (B, D, 9) 13 A, B, D

E (D, E, 2) 15 A, B, D, E

2 marks per correct row. [20 marks] [QUESTION Total 20]

Question 3: if Bubble Sort

perform the Bubble Sort algorithm on the array **[53, 42, 79, 65, 33, 24]**.

Pass 1:

* Comparing 53 and 42: 42 < 53, so swap them.
* Comparing 53 and 79: No swap needed.
* Comparing 79 and 65: 65 < 79, so swap them.
* Comparing 79 and 33: 33 < 79, so swap them.
* Comparing 79 and 24: 24 < 79, so swap them.

Array after Pass 1: **[42, 53, 65, 33, 24, 79]**

ꢀc University of Westminster Page 10

Pass 2:

* Comparing 42 and 53: No swap needed.
* Comparing 53 and 65: No swap needed.
* Comparing 65 and 33: 33 < 65, so swap them.
* Comparing 65 and 24: 24 < 65, so swap them.
* Comparing 65 and 79: No swap needed.

Array after Pass 2: **[42, 53, 33, 24, 65, 79]**

Pass 3:

* Comparing 42 and 53: No swap needed.
* Comparing 53 and 33: 33 < 53, so swap them.
* Comparing 53 and 24: 24 < 53, so swap them.
* Comparing 53 and 65: No swap needed.
* Comparing 65 and 79: No swap needed.

Array after Pass 3: **[42, 33, 24, 53, 65, 79]**

Pass 4:

* Comparing 42 and 33: 33 < 42, so swap them.
* Comparing 42 and 24: 24 < 42, so swap them.
* Comparing 42 and 53: No swap needed.
* Comparing 53 and 65: No swap needed.
* Comparing 65 and 79: No swap needed.

Array after Pass 4: **[33, 24, 42, 53, 65, 79]**

Pass 5:

* Comparing 33 and 24: 24 < 33, so swap them.
* Comparing 33 and 42: No swap needed.
* Comparing 42 and 53: No swap needed.
* Comparing 53 and 65: No swap needed.
* Comparing 65 and 79: No swap needed.

Array after Pass 5: **[24, 33, 42, 53, 65, 79]**

So, the array is sorted, and no more passes