UNIVERSITY OF COLOMBO, SRI LANKA

UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING

DEGREE OF BACHELOR OF INFORMATION TECHNOLOGY (EXTERNAL)

*Academic Year 2020 – 2nd Year Examination – Semester 4*

# *IT4105 – Programming II*

## *Part 2 - Structured Question Paper*

***(ONE HOUR)***

**To be completed by the candidate**

BIT Examination Index No: ..............................

**Important Instructions:**

- The duration of the paper is **1 (one) hour**.
- The medium of instruction and questions is English.
- This paper has **2 questions** and **9 pages**.
- **Answer all questions.**
- **Write your answers** in English using the space provided **in this question paper**.
- Do not tear off any part of this answer book.
- Under no circumstances may this book, used or unused, be removed from the Examination Hall by a candidate.
- Note that questions appear on both sides of the paper.
  If a page is not printed, please inform the supervisor immediately.
- Calculators are **not allowed**.
- *All Rights Reserved.*

**Questions Answered**

Indicate by a cross (×), (e.g. ☒ ) the numbers of the questions answered.

| | Question numbers | | |
|---|---|---|---|
| **To be completed by the candidate by marking a cross (×).** | 1 | 2 | |
| To be completed by the examiners: | | | |
| | | | |
| | | | |

1) (a) Consider the following array A.

A=[7 11 14 6 9 4 3 12]

(i) Show the result of running the partition subroutine of quicksort on the above array A, assuming that the index of the pivot is chosen to be 0 (the pivot is A [0] =7). What value does the partition return?

(5 Marks)

ANSWER IN THIS BOX

6 4 3 7 11 14 9 12

(ii) How many pivot values are needed to sort the above array A?
Note: You may use the first element as the pivot value in each sub-array.

(5 Marks)

ANSWER IN THIS BOX

5

(b) The following listing shows the incomplete Java code for the Quick Sort algorithm.

```
public void quickSort(int arr[], int begin, int end) {
    if (begin < end) {
        int partitionIndex = partition(arr, begin, end);
……………(i)……………………………………….
……………(ii)…………………………………...
……………}………………………………………
………(iii)………………………………………...
```

What are the suitable java commands/syntax for the blank positions in the code listing?

(4 Marks)

ANSWER IN THIS BOX

(i) quickSort(arr, begin, partitionIndex-1);

(ii) quickSort(arr, partitionIndex+1, end);

(iii) }

(c) Consider the follow array.

[38, -3, 47, 55, 1, 58, 16, 96, -84, 7]

If one uses the following sorting algorithms to sort the above data set in ascending order, what would be the intermediate step after the first iteration?

(i) Bubble sort,
(ii) Selection sort,
(iii) Insertion sort

(6 Marks)

| | |
|---|---|
| (i) | -3,38,47,1,55,16,58,-84,7,96 |
| | Or -84,38, -3,47,55,1,58,16,7 |
| (ii) | |
| | -84,-3,47,55,1,58,16,96,38,7 |
| | Or 38, -3,47,55,1,58,16,7, -84,96 |
| (iii) | -3, 38 47, 55, 1, 58, 16, 96, -84, 7 |
| | Or 38, -3, 47, 55, 1, 58, 16, 96, -84, 7 |

(d) Consider the following Binary Search Algorithm.

1. *Compare x with the middle element.*
2. *If x matches with middle element, we return the middle index.*
3. *Else If x is greater than the middle element, then x can only lie in right half subarray after the middle element. So, we recur for right half.*
4. *Else (x is smaller) recur for the left half.*

Note: You may assume x as the search (target) element.

If one uses the following data set to search **25** (value of the x) using the binary search algorithm, how many comparisons are needed to find the search (target) value.

**Date Set =[ 10,15,25,36,45,82,95,101,110]**

**Hint**: you may use the following formula when calculating the middle value. `middle= (lower bound index of the array+ upper bound index of the array)/2` and take the whole integer value ignoring the decimal portion.

(5 Marks)

3 comparisons (45,15 and including 25)

2)(a) You have been given instructions to create an Adelson, Velski & Landis (AVL) tree using the following set of integer values.

5,22,66,11,9,34,14,12,

During the AVL construction process, imbalance situations may occur. Fill in the table below for columns 2 and 3 of such imbalance scenarios.

(6 Marks)

| Newly inserted Node | Node at which the imbalance occurred | Name the rotation required to restore the balance |
|---|---|---|
| 5 | | |
| 22 | | |
| 66 | 5 | Single Rotation |
| 11 | | |
| 09 | 5 | Double Rotation |
| 34 | | |
| 14 | | |
| 12 | 11 | Double Rotation |

(b) A programmer creates a Binary Search Tree with the data set given below and subsequently a node is deleted from the tree. If this node could be one of the following 92, 35, 29, and 20, determine the suitable node(s) from the remaining nodes to replace the deleted node. You may assume that every deletion is independent and would be performed on the original tree.

Data Set: 20,35,18,43,92,29,27

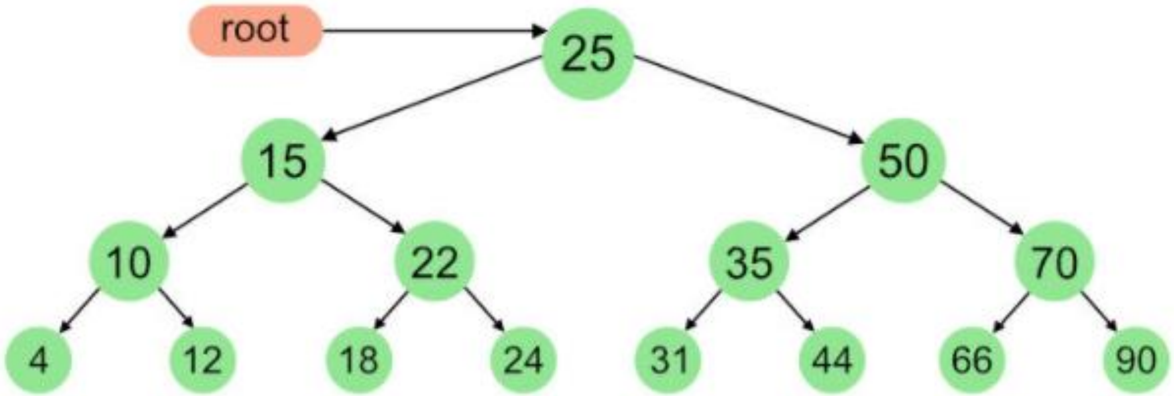
Nodes to be deleted are:
(i) 92
(ii) 35
(iii) 29
(iv) 20

(4 Marks)

**ANSWER IN THIS BOX**

| Deleted Node | Replacing Node(s) |
|---|---|
| (i)92 | No replacing node, deleted node is a leaf |
| (ii)35 | 29 or 43 |
| (iii)29 | 27 |
| (iv)20 | 18 or 27 |

(c) Consider the following Binary tree.



What is the (i) In-order (ii) Pre-Order and (iii) Post-order traversal of the above tree.

(6 Marks)

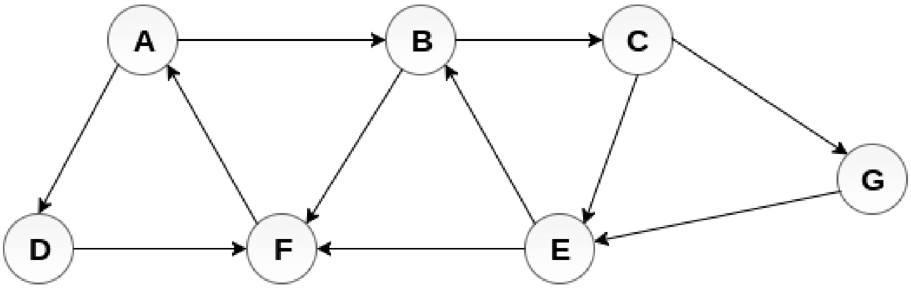
**ANSWER IN THIS BOX**

(i)4,10,12,15,18,22,24, 25,31,35,44,50,66,70,90

(ii)25,15,10,4,12,22,18, 24,50,35,31,44,70,66,90

(iii)4,12,10,18,24,22,15,31,44,35,66,90,70,50,25

(d) Consider the following directed graph.



(i) Write down the adjacency list representation of the above graph.

(4 Marks)

**ANSWER IN THIS BOX**

Adjacency List

| Nodes | Adjacent Nodes |
|---|---|
| A | B, D |
| B | C, F |
| C | E, G |
| G | E |
| E | B, F |
| F | A |
| D | F |

(ii) A programmer wants to traverse the graph using the breadth-first search (BFS) algorithm. Explain the implementation procedure of the algorithm using two queues.

(5 Marks)

**ANSWER IN THIS BOX**

**Let’s start examining the graph from Node A.**

1. Add A to QUEUE1 and NULL to QUEUE2
   QUEUE1 = {A}
   QUEUE2 = {NULL}

2. Delete the Node A from QUEUE1 and insert all its neighbours. Insert Node A into QUEUE2
   QUEUE1 = {B, D}
   QUEUE2 = {A}

3. Delete the node B from QUEUE1 and insert all its neighbours. Insert node B into QUEUE2.
   QUEUE1 = {D, C, F}
   QUEUE2 = {A, B}

4. Delete the node D from QUEUE1 and insert all its neighbours. Since F is the only neighbour of it which has been inserted, we will not insert it again. Insert node D into QUEUE2.
   QUEUE1 = {C, F}
   QUEUE2 = { A, B, D}

5. Delete the node C from QUEUE1 and insert all its neighbours. Add node C to QUEUE2.

   QUEUE1 = {F, E, G}
   QUEUE2 = {A, B, D, C}

6. Remove F from QUEUE1 and add all its neighbours. Since all of its neighbours has already been added, we will not add them again. Add node F to QUEUE2.
   QUEUE1 = {E, G}
   QUEUE2 = {A, B, D, C, F}

7. Remove E from QUEUE1, all of E's neighbours has already been added to QUEUE1 therefore we will not add them again. Add node E to QUEUE2
   QUEUE1 = {G}
   QUEUE2 = {A, B, D, C, F, E}

8. Remove G from QUEUE1, all of G's neighbours has already been added to QUEUE1 therefore we will not add them again. Add node G to QUEUE2
   QUEUE1= {Null}
   QUEUE2 = {A, B, C, D, F, E, G} OR other possibilities.
   Traversal order is QUEUE2 = {A, B, C, D, F, E, G} OR other possibilities.

**************************************************