The Open University of Sri Lanka
Faculty of Engineering Technology
Department of Electrical and Computer Engineering

042

| | |
|---|---|
| Study Programme | : Bachelor of Technology Honours in Engineering |
| Name of the Examination | : Final Examination |
| Course Code and Title | : **EEX4435 Data Structures and Algorithms** |
| Academic Year | : 2022/23 |
| Date | : 05th February 2024 |
| Time | : 09:30 – 12:30 hrs. |
| Duration | : **3 hours** |

**General Instructions**

1. Read all instructions carefully before answering the questions.
2. This question paper consists of **five (5)** questions in **four (4)** pages.
3. Answer all **FIVE (5)** questions. All questions carry equal marks.
4. Answer for each question should commence from a new page.
5. Answers should be in clear handwriting and do not use Red colour pen.
6. Clearly state your assumptions, if any.
7. This is a **Closed Book Test** (CBT).

**Q1**

(a) Find the time complexity (Big-Oh) for the following recurrence relations. **Show all the steps.** [09]

i) $X(n) = X(n - 1) + 5$, for $n > 1$, $X(1) = 0$.

ii) $X(n) = 3*X(n - 1)$, for $n > 1$, $X(1) = 4$.

iii) $X(n) = X(n - 1) + n$, for $n > 0$, $X(0) = 0$.

(b) Answer the following questions for the algorithm (pseudo code) given below:

```
Algorithm Sample (A[0..n - 1, 0..n - 1])
// Input: The matrix A_{n×n} of real numbers
for i ← 0 to n – 2 do
    for j ← i + 1 to n - 1 do
        if A[i, j] ≠ A[j, i]
            return false
return true
```

i) What does this algorithm compute? [03]

ii) What is its basic operation? [02]

iii) How many times is the basic operation executed (best-case and worst-case, separately)? [04]

iv) What is the overall time complexity of this algorithm? [02]

**Q2**

(a) Consider the following queue of characters. This queue is a circular array which is allocated six memory cells as shown below (it has drawn as a linear array). The symbols *f* and *r* indicate the *front* and *rear* pointers of the array, respectively.

Current status of the Queue:

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| | A | C | D | | |
| | *f* | | *r* | | |

Redraw the queue as above after each operation given below, sequentially**. You should indicate the location of the *front* and *rear* pointers. [10]

i) 'Y' is added to the queue;

ii) Two items are removed;

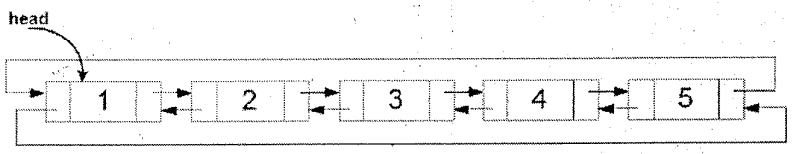
iii) 'K', 'L' and 'M' are added, respectively;

iv) Two letters are deleted;

v) 'Z' is added to the queue.

** You need to follow the order, i.e., i) → ii) → . . . → v) in both parts: (a) and (b).

(b) The circular doubly linked list with the head pointer pointing to the first node in the list is shown below. The previous and next links are represented by 'prev' and 'next', respectively.




Redraw the whole linked list (clearly showing the links which have been re-adjusted), after each of the following operations changes are made, sequentially**. [10]

i) head→next→next→next = head→prev;

ii) head→prev→prev→prev = head→next→next→next→prev;

iii) head→next→next→next→prev = head→prev→prev→prev;

iv) head→next = head→next→next;

v) head→next→prev→next = head→next→next→next;

**Q3** Consider the following operands and operators with their precedence for logic expressions. Note that you must consider left and right parenthesis as well.

| Operands |
|---|
| 0<br>1 |

| Operators and their meaning | | Precedence |
|---|---|---|
| ~ | not | First |
| & | and | Second |
| \| | or | Third |

(a) Write the pseudocode algorithm to convert a string of logic expression into its expression tree. You can use any data structure to represent the tree and should explain how your data structure logically relates to the tree. Since ~ is a unary operator, you should put its only operand to its right child. [12]

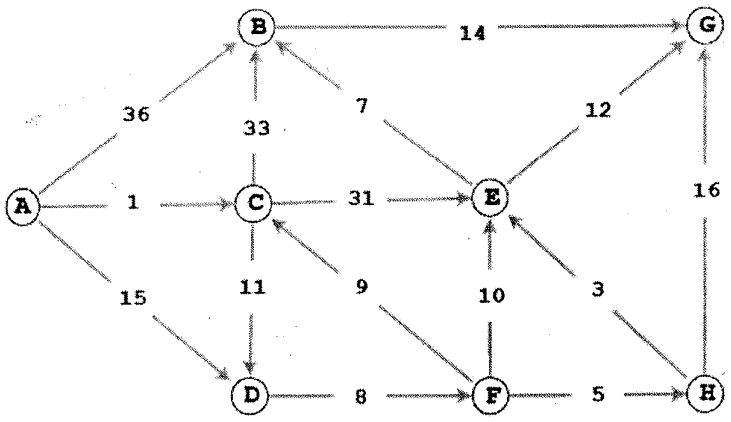
(b) Draw the expression tree of the following logical expression: [04]

$$\sim (0 \,\&\, \sim 1 \,\&\, 0 \mid 0) \,\&\, 0$$

(c) Obtain the prefix and postfix notations for the given expression in (b). [04]

**Q4**

Consider the following directed, weighted graph to answer the questions below.




(a) Step through Dijkstra's algorithm to calculate supposedly shortest paths from **A** to every other vertex. Show your steps in the table form. Cross out old values and write in new ones, from left to right, as the algorithm proceeds. [16]

(b) Briefly describe two limitations of using Dijkstra's algorithm to find shortest paths. [04]

**Q5**

(a) Insert the objects with key values {8, 15, 23, 5, 17, 3}, respectively, into an initially empty hash table of size 7 with the hash function $h(x) = x$ MOD 7, by using the following methods. Show the resulting table for each method.

i) Open addressing with linear probing; [04]

ii) Closed addressing with direct chaining; [04]

(b) Consider an implementation of a hash table with linear probing as given below:

> It uses an array, "***storage***" to keep pointers to the keys, and uses a Boolean array, "***used***" to record which locations in "***storage***" have contained a "***key***".

Write the function, "***lookup***( . . . )" in pseudocode for this hash table implementation, searching for a given "***key***", and returning (as a Boolean) whether it was found. [12]

-- End –