Universiti Tun Hussein Onn Malaysia

UNIVERSITI TUN HUSSEIN ONN MALAYSIA

FINAL EXAMINATION
SEMESTER II
SESSION 2023/2024

COURSE NAME	:	DATA STRUCTURES AND ALGORITHMS
COURSE CODE	:	BEJ 32103
PROGRAMME CODE	:	BEJ
EXAMINATION DATE	:	JULY 2024
DURATION	:	2 HOURS 30 MINUTES
INSTRUCTIONS	:	1. ANSWER ALL QUESTIONS. 2. THIS FINAL EXAMINATION IS CONDUCTED VIA ☒ Open book ☐ Closed book 3. STUDENTS ARE **NOT PROHIBITED** TO CONSULT THEIR OWN MATERIAL OR ANY EXTERNAL RESOURCES DURING THE EXAMINATION CONDUCTED VIA OPEN BOOK.

THIS QUESTION PAPER CONSISTS OF **SIX (6)** PAGES

Q1 Answer (a) and (b) based on **Figure Q1.1**.

```
Algorithm dHigh (k, A)
//Input: An integer-type array A with k size.
//Output: The largest element in the array A
        int high, i;              //Line1
        high = A[0];              //Line2
        i=1;                      //Line3
        while (i<k) {             //Line4
            if (high < A[i])      //Line5
                high = A[i];      //Line6
            i++;                  //Line7
        }                         //Line8
        return high;              //Line9
```

Figure Q1.1 Algorithm of *dHigh*

(a) Determine the running time in the Big-Oh notation for **Figure Q1.1**. Include steps and procedures to obtain your answer.

(15 marks)

(b) Referring to **Figure Q1.1**, provide a brief explanation of how the best-case and worst-case scenarios can occur. For each scenario, provide a justification for your answer.

(5 marks)

Q2 (a) Define a stack in the data structure.

(2 marks)

(b) State two methods to implement the stack.

(2 marks)

(c) State the stack operations for the following events in sequence order.

Fill in the empty stack with 10, 5, 12, 20, 7 and 1.

Then, peek the stack. Finally, remove 12 from the stack.

(6 marks)

Q3 **Figure Q3.1** shows the algorithm to add elements in queue *C*.

```
Algorithm addElement()
//x represents element in queue A
//y represents element in queue B
        C = createQueue
        i = 0
        loop (not empty A AND not empty B)
            i = i + 1
            dequeue (A, x)
            dequeue (B, y)
            if (y NOT i)
                enqueue (C, x)
```

Figure Q3.1 Algorithm of *addElement*

The contents of queue *A* and *B* are given as follows. Note that the queue contents are shown from front (left) to rear (right).

A: 42 30 41 31 19 20 25 14 10 11 12 15

B: 1 4 5 4 10 13

(a) Illustrate the execution of the algorithm in **Figure Q3.1** by using **Table Q3.1**.

Table Q3.1 Execution of the algorithm using tracing table

Loop (not empty A AND not empty B)	i=i+1	dequeue (A,x)	dequeue (B,y)	y NOT i	enqueue (C,x)

(8 marks)

(b) State the final content of queue *C*.

(2 marks)

Q4 The algorithm in **Figure Q4.1** is intended to add an element at the beginning of a linked list when the list is not empty. However, if the list is empty, the new element cannot be added using the following algorithm.

```
Algorithm addNewElement()
Begin
    1. Create a new node.
    2. Insert the new element to the new node.
    3. Make the new node points to the head.
    4. Update head to point to the new node.
End
```

Figure Q4.1 Algorithm of *addNewElement*

Therefore, change the algorithm of **Figure Q4.1** so that it can add an element to the beginning of the linked list, regardless of whether the list is empty or not.

(15 marks)

Q5 (a) Construct an algorithm to insert new nodes into a tree using *Binary Search Tree* (BST).

(5 marks)

(b) Insert seven new nodes using BST to the tree in **Figure Q5.1**. The sequence of insertion is as follows: add node 0019, follows with node 0012, then node 0025, 5230, 0043, 0090, and finally, node 0006.

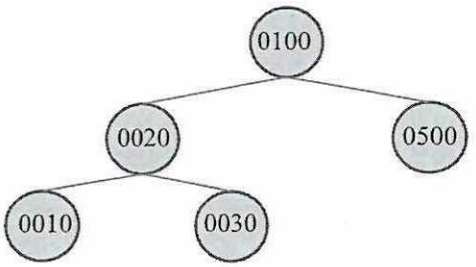

Figure Q5.1 *Binary Search Tree*

Show the final tree after all new nodes are inserted.

(15 marks)

Q6 (a) Determine whether the binary tree illustrated in **Figure Q6.1** is a complete binary tree. Justify your reason.

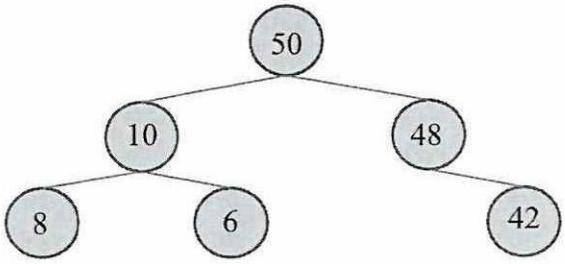

Figure Q6.1 *Binary tree*

(2 marks)

(b) Show the resulting heap after the *reheap down* is performed when the root of the heap in **Figure Q6.2** is removed. Include steps to get the result.

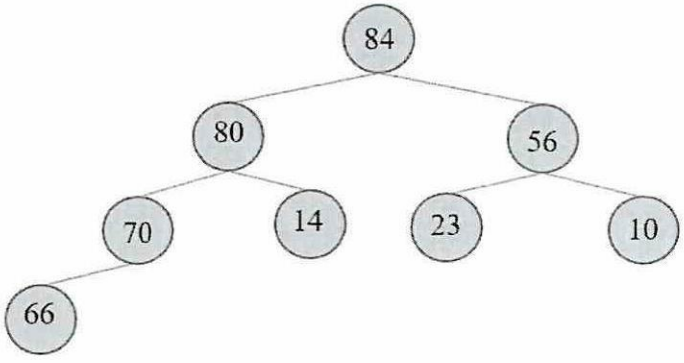

Figure Q6.2 *Heap*

(3 marks)

(c) Produce the shortest path diagram from node *J* to all other nodes in the **Figure Q6.3** using Dijkstra's algorithm.

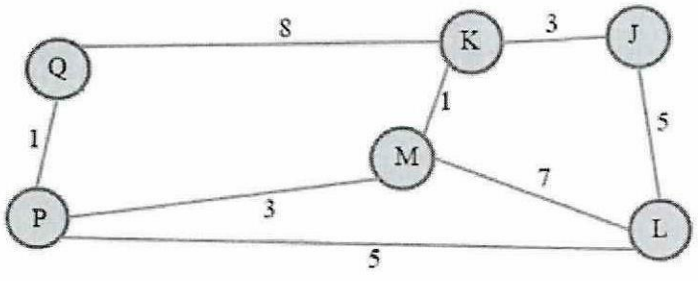

Figure Q6.3 *Graph*

(10 marks)

Q7 Given partial code of sorting algorithm in the **Figure Q7.1**. Note that *n* is an array size.

```
for (i=0; i<=n-2; i++)  {      // Line1
        if (Z[i]<Z[i+1]) {     // Line2
        t=Z[i];                // Line3
        Z[i]=Z[i+1];           // Line4
        Z[i+1]=t;              // Line5
        } }                    // Line6
```

Figure Q7.1 Partial code of sorting algorithm

Initially, the element of array *Z* is stored as shown in **Figure Q7.2**.

Z	10	23	2	12	34
	[0]	[1]	[2]	[3]	[4]

Figure Q7.2 Initial elements of array *Z*

(a) Does the array *Z* store the data as illustrated in **Figure Q7.3** after the execution of **Figure Q7.1**?

Z	23	10	12	2	34
	[0]	[1]	[2]	[3]	[4]

Figure Q7.3 Final elements of array *Z*

(2 marks)

(b) Use **Table Q7.1** as a tool to validate your answer in **Q7(a)**.

Table Q7.1 Validation using the trace table

Line1 i<=n-2	Line2	Line3	Line4	Line5	Line1 i++

(8 marks)

- END OF QUESTIONS -