CM2035

BSc EXAMINATION

COMPUTER SCIENCE

Algorithms and Data Structures II

Release date: Tuesday 7 March 2023 at 12:00 midday Greenwich Mean Time

Submission date: Wednesday 8 March 2023 by 12:00 midday Greenwich Mean Time

Time allowed: 24 hours to submit

INSTRUCTIONS TO CANDIDATES:

Section A of this assessment paper consists of a set of **TEN** Multiple Choice Questions (MCQs) which you will take separately from this paper. You should attempt to answer **ALL** the questions in Section A. The maximum mark for Section A is **40**.

Section A will be completed online on the VLE. You may choose to access the MCQs at any time following the release of the paper, but once you have accessed the MCQs you must submit your answers before the deadline or within **4 hours** of starting, whichever occurs first.

Section B of this assessment paper is an online assessment to be completed within the same 24-hour window as Section A. We anticipate that approximately **1 hour** is sufficient for you to answer Section B. Candidates must answer **TWO** out of the THREE questions in Section B. The maximum mark for Section B is **60**.

Calculators are not permitted in this examination. Credit will only be given if all workings are shown.

You should complete **Section B** of this paper and submit your answers as **one document**, if possible, in Microsoft Word or a PDF to the appropriate area on the VLE. Each file uploaded must be accompanied by a coversheet containing your **candidate number** written clearly at the top of the page before you upload your work. Do not write your name anywhere in your answers.

© University of London 2023
UL23/0214

SECTION A

Candidates should answer the **TEN** Multiple Choice Questions (MCQs) quiz, **Question 1** in Section A on the VLE.

SECTION B

Candidates should answer any **TWO** questions in Section B.

Question 2

This question is about Binary Search Trees (BSTs).

(a) The numbers 4, 5, 3, 2, 6, in that order, are inserted into an empty BST. Draw the resulting BST.

[5 marks]

(b) Consider

```
function F1(root)
    if (root != null)
        F1(root → left)
        print(root.key)
        F1(root→right)
end function
```

What is printed when F1 is run on the root of the BST in part (a)?

[5 marks]

(c) F2, below, inserts a node n in a BST with root x. What is condition C?

```
function F2(Node x, Node n)
    y = null
    while (x != null)
        y = x
        if (n→key < x→key)
            x = x→left
        else
            x = x→right
    if (C)
        y→left = n
    else
        y→right = n
end function
```

[5 marks]

(d) A BST has N nodes. Use theta notation to express the worst-case search time complexity. Justify your answer.

[5 marks]

(e) Consider

root: the root of a binary search tree (BST) storing positive integer numbers greater than 0
N: the number of nodes in the BST rooted at root
floor(x): returns the largest integer smaller than or equal to x.

```
function F3(root,N)
    if(N == 0):
        return -1
    A=new array(N) of zeroes
    Q=new Queue()
    ENQUEUE(Q,root)
    while !ISEMPTY(Q)
        x=PEEK(Q)
        A[0]=x→data
        i=0
        while A[i]>A[i+1] and i<N-1
            t=A[i]
            A[i]=A[i+1]
            A[i+1]=t
            i=i+1
        ENQUEUE(Q,x→left)
        ENQUEUE(Q,x→right)
        DEQUEUE(Q)
    return A[floor(N/2)]
end function
```

What is the purpose of F3?

[5 marks]

(f) What, in big-O notation, is the worst-case time complexity of F3 in part (e) for a BST if N nodes. Explain your reasoning.

[5 marks]

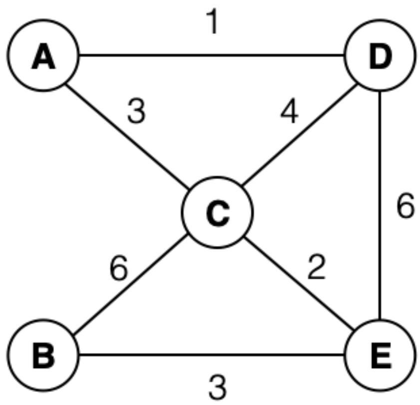
Question 3

This question is about graphs.

(a) When is it better to use an adjacency list to represent a graph rather than using an adjacency matrix? Explain your reasoning.

[5 marks]

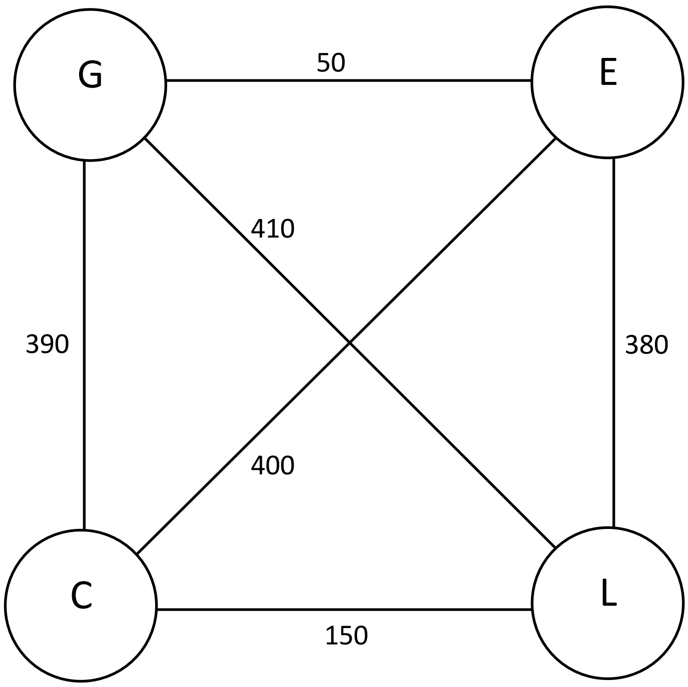
(b) Consider the following weighted, undirected graph:



By hand, go through an implementation of Dijkstra's algorithm on this graph to find the shortest path between nodes A and B. Your implementation should produce both the path's length and the nodes in it.

[10 marks]

(c) The traveling salesperson problem (TSP) can be stated as: find the minimum-distance tour of N cities such that each city is visited once, and the tour ends at the starting city. The problem can be represented as a fully connected graph with non-directed weighted edges where the vertices are cities and edge weights are the distances between cities. For example, the graph



shows road distances (rounded to the nearest ten miles) between Glasgow, Edinburgh, Cardiff and London. The tour GCELG has distance 390 + 400 + 380 + 410 = 1580 miles.

(i) Write a pseudocode algorithm that will search for an approximate optimal tour of an N-city TSP.

[5 marks]

(ii) Consider an algorithm that finds an optimum tour by exhaustive search. What is the time-complexity of this algorithm?

[5 marks]

(iii) With regard to your answer to (ii) above, and by comparison to the time complexities of algorithms you have studied in this module, what are the prospects for discovering a polynomial time solution to the TSP?

[5 marks]

Question 4

This question is about hashing.

(a) Briefly explain why hash tables are useful for storing many integers.

[5 marks]

(b) Explain how hash functions are used to build a hash table. Ignore collisions and illustrate your answer by considering a function $h(k) = k \bmod 5$ hashing into a table with 5 slots.

[5 marks]

(c) Briefly explain the concept of a collision in the context of a hash function. Give an example demonstrating this concept.

[5 marks]

(d) Suppose that T is table with m slots and $h(k, i)$ for $I = 0, 1, \ldots, m$ is a hash function that hashes k to $j = h(k, 0)$ if slot j of T is unoccupied and otherwise probes T for an available slot. Write a pseudocode algorithm INSERT(T, k) that attempts to insert k in table T. INSERT should return the slot index if the insertion was successful; an error warning should be returned if insertion fails.

[5 marks]

(e) Explain why linear probing can lead to long runs of occupied slots.

[5 marks]

(f) Suggest a general strategy to overcome the pathology of part (e) and provide an example of a specific hash function that ameliorates the problem.

[5 marks]

END OF PAPER