No calculator allowed in this examination

UNIVERSITY OF BIRMINGHAM

School of Computer Science

Data Structures, Algorithms and Databases

Main Summer Examinations 2024

Time allowed: 2 hours

[Answer all questions]

Note

Answer ALL questions. Each question will be marked out of 20. The paper will be marked out of 80, which will be rescaled to a mark out of 100.

Question 1 BST and AVL

Let $\_array = [20, 35, 5, 10, 18, 14, 35, 60, 75]$. Consider a function named `arrayToBST` that takes *_array* as the input and inserts the elements of *_array* in a **BST** *tree* (without rebalancing).

```
function arrayToBST(_array)
  tree = □;
  for element in _array
    tree = insert (element,tree);
  return tree;
```

(a) Draw the resulting tree. **[2 marks]**

(b) What is the height of the tree? **[2 marks]**

(c) Is it an AVL tree? Justify your answer. **[2 marks]**

(d) Consider a function named `arrayToAVL` that inserts the elements of *_array* in an **AVL** *tree*. The *insert_withRebalance*() function will perform a rebalance if required.

```
function arrayToAVL(_array)
  tree = □;
  for element in _array
    tree = insert_withRebalance (element,tree);
  return tree;
```

Use this function on $\_array = [20, 35, 5, 10, 18, 14, 35, 60, 75]$, show the results. Illustrate the intermediate stages (before and after each rebalance) as well as the final tree. **[8 marks]**

(e) Here is a purported *rebalance* function that takes a *_node* as the input.

Assume that *_node.left* is the left subtree of a node; *_node.right* is the right subtree of a node; *rightRotation*() is a function that performs a left-to-right single rotation at a given node and *leftRotation*() similarly does a right-to-left single rotation.

```
function rebalance(_node)
  if _node has LL case imbalance
    return leftRotation(_node.right)

  if _node has RR case imbalance
    return rightRotation(_node.left)

  if _node has LR case imbalance
    _node.right = rightRotation(_node.left)
    return leftRotation(_node)

  if _node has RL case imbalance
    _node.left = leftRotation(_node.right)
   return rightRotation(_node)

  return _node
```

There are errors/bugs in the above pseudo-code which lead to a wrong rebalance result. Write the corrected pseudo-code in the same style. **[6 marks]**

This page intentionally left blank.

Question 2 Entity-Relationship modelling

A database is required for an airline flight reservation system. This system enables customers to book flights by selecting the flight number, origin, destination, departure date, and time. Each booking is assigned a unique booking ID number and contains booking for multiple passengers. The airline allocates baggage allowance for the booking, in terms of the number of pieces of baggage which may depend on the frequent flyer status of the customer.

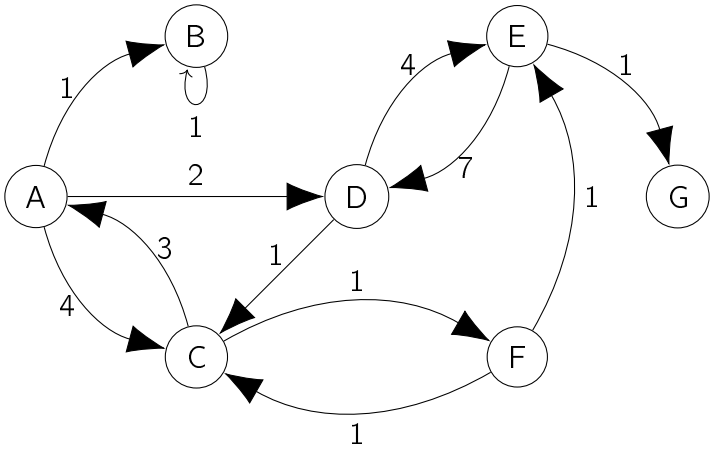
(a) Develop an Entity-Relationship model for the application. Explain the key design decisions made in your choice of entities, relationships and any other (ownership or hierarchical) aspects. Annotate it with multiplicities and hierarchy annotations, as required. **[10 marks]**

(b) Carry out logical design for the model, representing the design with relational schemas for tables. Annotate the schemas with primary keys and the possibility of null attributes. **[5 marks]**

(c) Write CREATE TABLE statements for 2–3 tables. Include among them a table that represents a relationship or incorporates a relationship, a weak entity or a subclass entity. The other table(s) may be those that are linked to this table. **[5 marks]**

Question 3 Graphs and Max-Heap Trees

(a) Consider the following weighted directed graph:



Calculate the **shortest path** from *A* to *G* using the Dijkstra's algorithm. ("Shortest" means the path with the lowest total weight.) **[10 marks]**

You are expected to show your work using a table of the following form and also list the shortest path (e.g. A→B→C) and and specify the resulting weight:

A	B	C	D	E	F	G	Finished
0,A	∞,B	∞,C	∞,D	∞,E	∞,F	∞,G	

Total Weight:
Shortest Path:

Consider the following algorithm and let n be the length of array.

```
def insert_all(array):
  d = <empty data structure>
  for x in array:
    <insert x in data structure d>
```

(b) Suppose `d` is a binary search tree, without balancing. Give the time complexity of `insert_all` as a function of n in "big-Oh" or $O(\ )$ notation. **[3 marks]**

(c) Suppose `d` is a hash table. Give the time complexity of `insert_all` as a function of n in "big-Oh" or $O(\ )$ notation. **[3 marks]**

(d) Does there exist an input array for which `insert_all` has same time complexity, regardless of whether `d` is a binary search tree or `d` is a hash table? If the answer is yes, give an example array of length $n = 7$. If the answer if no, motivate your answer. **[4 marks]**

Question 4 Relational algebra and normalisation

Consider the following schemas for a database used by a student registration system.

Table course	
cid	integer
title	varchar(40)
level	integer

Table prerequisite	
pre	integer
post	integer

Table student	
sid	integer
name	varchar(40)

Table registered	
sid	integer
cid	integer
year	integer

The table *prerequisite* describes prerequisites for courses, indicating that a "*pre*" course must be taken before taking a "*post*" course (Both "*pre*" and "*post*" are course ids). For example a course called "Programming 1" might be a prerequisite for a course called "Programming 2". In general, a course may have any number of prerequisites.

The table *registered* describes which students have registered for which courses in a particular year.

(a) Write a relational algebra expression for the following problem:

Find the ID numbers of students who took some level 2 course in 2023. **[2 marks]**

(b) Write a relational algebra expression for the following problem:

Find the ID numbers of students who took only level 2 courses in 2023.

Or, to put another way, find the ID numbers of students who registered for courses *all* of which are at level 2. **[4 marks]**

Hint: You may find the following logical equivalence useful in devising your solution:

$$\text{for all } x,\ P \iff \text{not (exists } x \text{ such that not } P)$$

(c) Explain in words what the following expression calculates:

$\pi_{\text{pre,post}}(\rho_{(\text{pre,cid})}(\text{prerequisite}) \bowtie \rho_{(\text{cid,post})}(\text{prerequisite}))$ **[4 marks]**

(d) Define precisely when a set of tables is said to be in Boyce-Codd normal form. **[4 marks]**

(e) Consider an employee table with the schema:

employee(*last_name*, *first_name*, *office*, *phone_number*)

and the functional dependency *phone_number* $\longrightarrow$ *office*.

Explain the possibility of **update anomaly** and **deletion anomaly** arising from this dependency, giving examples of situations that would lead to the anomalies. **[4 marks]**

How does the use of **Boyce-Codd normal form** avoid these anomalies? **[2 marks]**

Do not complete the attendance slip, fill in the front of the answer book or turn over the question paper until you are told to do so

Important Reminders

- Coats/outwear should be placed in the designated area.
- Unauthorised materials (e.g. notes or Tippex) must be placed in the designated area.
- Check that you do not have any unauthorised materials with you (e.g. in your pockets, pencil case).
- Mobile phones and smart watches **must** be switched off and placed in the designated area or under your desk. They must not be left on your person or in your pockets.
- You are not permitted to use a mobile phone as a clock. If you have difficulty seeing a clock, please alert an Invigilator.
- You are not permitted to have writing on your hand, arm or other body part.
- Check that you do not have writing on your hand, arm or other body part – if you do, you must inform an Invigilator immediately
- Alert an Invigilator immediately if you find any unauthorised item upon you during the examination.

Any students found with non-permitted items upon their person during the examination, or who fail to comply with Examination rules may be subject to Student Conduct procedures.