Calculators may be used in this examination provided they are not capable of being used to store alphabetical information other than hexadecimal numbers

UNIVERSITY OF BIRMINGHAM

School of Computer Science

Data Structures and Algorithms

Mock Exam 2026

Time allowed: 2 hours

[Answer all questions]

Note

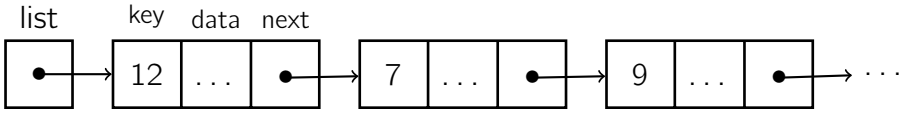
Answer ALL questions.

Question 1 Sorting Long Records

A multinational corporation is storing a large amount of data in a simply linked list. Each record has an index `key`, a pointer `next` to the next record, and a large amount of additional data of unspecified structure. Assume that a record is an instance of a Java class `Record` with the following structure:

```
class Record {
    int key;
    ...
    Record next;
}
```

(a) (i) Assume we want to swap the first two entries in the following list:



Draw a diagram after the swap, and clearly indicate which pointers need to change and how. **[3 marks]**

(ii) Write a function `swapTwo(Record list)` that takes *any* given list that has at least 2 records, swaps the first two records, and then returns the pointer to the new head of the list. Write all the pointer operations needed for this operation. **[5 marks]**

(b) Given two such lists, `list1` and `list2`, whose keys are sorted in increasing order, describe an algorithm that merges these two lists into a single list whose keys are in increasing order and returns a pointer to the head of the new list. Use pseudocode to answer this question. **[12 marks]**

(c) Describe an algorithm using pseudocode that sorts such a list in time $O(n \log n)$ where n is the number of records, and give a brief justification of the time complexity. **[12 marks]**

Question 2 AVL Trees

You are given the following sequence of numbers that need to be inserted into an initially empty AVL tree:

$$5, 3, 1, 7, 6, 9, 2, 8, 4.$$

(a) Insert the numbers in the given order (left-to-right) while maintaining AVL tree balance. Draw the AVL tree after each insertion; at each step, if a rotation was performed write the reason for this rotation. **[10 marks]**

(b) Rearrange the given sequence so that constructing the AVL tree as above does not require any rotations. Draw the final AVL tree (intermediate steps do not need to be drawn in this case). **[6 marks]**

(c) Assume we are given a sequence of n numbers (where $n > 2$ and n is odd) in strictly increasing order and we insert them in the given order to an initially empty AVL tree. How many total rotations are required in the process? Express your answer using the big-O notation.

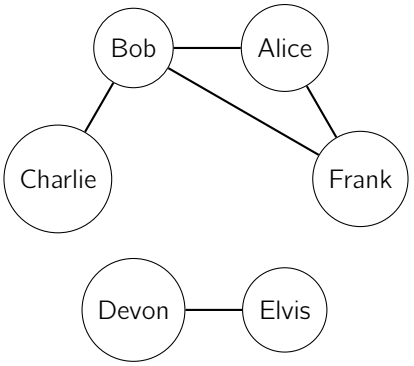
Hint: Try a few insertions to discover a pattern. **[6 marks]**

(d) From the lectures, we know that, in an AVL tree, the *balance* at every node is either $1, 0$, or -1. Now consider an *almost-AVL tree* (a more relaxed version of an AVL tree) where the balance at every node can be $-2, -1, 0, 1$, or 2. Derive an upper bound on the height of an almost-AVL tree with n nodes; express the result using the big-O notation. **[10 marks]**

Question 3 A Day in Life of a Social Network Engineer

A social network designed by The Social Network Company is modelled as an undirected graph $G = (V, E)$, with $|V| = n$ vertices and $|E| = m$ edges. The vertices represent people, while edges represent friendships. We say two people are *friends* if an edge directly connects them, and we say that two people are *connected* if there is a path of edges connecting them. The *distance* between two connected people is the number of edges between them.

For example, in the below graph, Alice and Bob are friends, Bob and Charlie are friends, and Devon and Elvis are friends; Alice is connected to Bob, Charlie, and Frank, but not Devon or Elvis; Alice's distance to Bob is 1, but Alice's distance to Charlie is 2.



Assume that the graph is given by lists of neighbours, i.e., as a number of vertices `n` and a two-dimensional array `int[][] friends` where `friends[v]` is the list of of all friends of vertex `v`. (If `u` and `v` are friends, then `u` is in `friends[v]` and `v` is in `friends[u]`.)

(a) Describe an algorithm (using pseudocode) which takes a two-dimensional array (as described above) and an integer k, and returns a list of integers which each correspond to a person who has at least k friends. What is the time complexity of your algorithm, in terms of n and m? **[6 marks]**

(b) An influencer recently released a broadcast message which must be sent from the influencer to all other persons connected to them as quickly as possible. The message is sent in rounds, where in each round, anyone who receives the message can forward it to any number of their friends (or do nothing if no message is forwarded to them).

Here is an algorithm for getting the message to everyone connected to the influencer: In any round, whenever a person receives the broadcast message for the first time from some of their friends, they communicate the message to the rest of their friends. The algorithm terminates when everyone has received the message. This is known as *flooding the network*.

(i) What is the number of rounds of this algorithm in the worst-case? Express the result as $\Theta(T)$, and justify your answer by giving an example of a network where you need $\Omega(T)$ rounds. **[6 marks]**

(ii) You have discovered that the maximum distance of any person from the influencer is 10. In light of this new information, what is the number of rounds needed in the worst-case of the flooding algorithm? Justify your answer.
[4 marks]

(c) As an engineer at The Social Network Company, you are asked to check if every two people in a given network are connected. What is the most efficient algorithm to check that? Describe the algorithm using pseudocode. What is its time complexity?
[10 marks]

(d) You discover that the social network contains many *clusters*. A *cluster* is a group of people who are all connected; two different clusters have no edge between them. For example, the above social network has two clusters: one connecting Alice, Bob, Charlie, and Frank; and the other connecting Devon and Elvis.

How would you use graph traversal techniques to count the number of such clusters? Describe the algorithm using pseudocode. What is its time complexity?
[10 marks]

End of Paper