EECS 484 Sample Midterm Exam 2 (Question Book)

1. This is a closed book exam. But you are allowed to bring hand-written notes on one double-sided 8.5x11 sheet of paper with you.
2. Write your login ID (uniqname) and name on this question book and the answer book.
3. Write your answers in the answer book.
4. You have 120 minutes to complete this exam.
5. If you see typos that are confusing, ask us to clarify. If a problem is ambiguous and you don't have time to clarify, state the assumptions and answer the problem.
6. No electronic devices are allowed, including calculators, smartphones, computers, etc. Please power them down and place in your backpack.
7. Please sign the honor pledge, turn in this exam, and show a picture ID when turning in the exam to a member of the teaching staff. Thank you.

Your exam room number: _____

Your name: _____

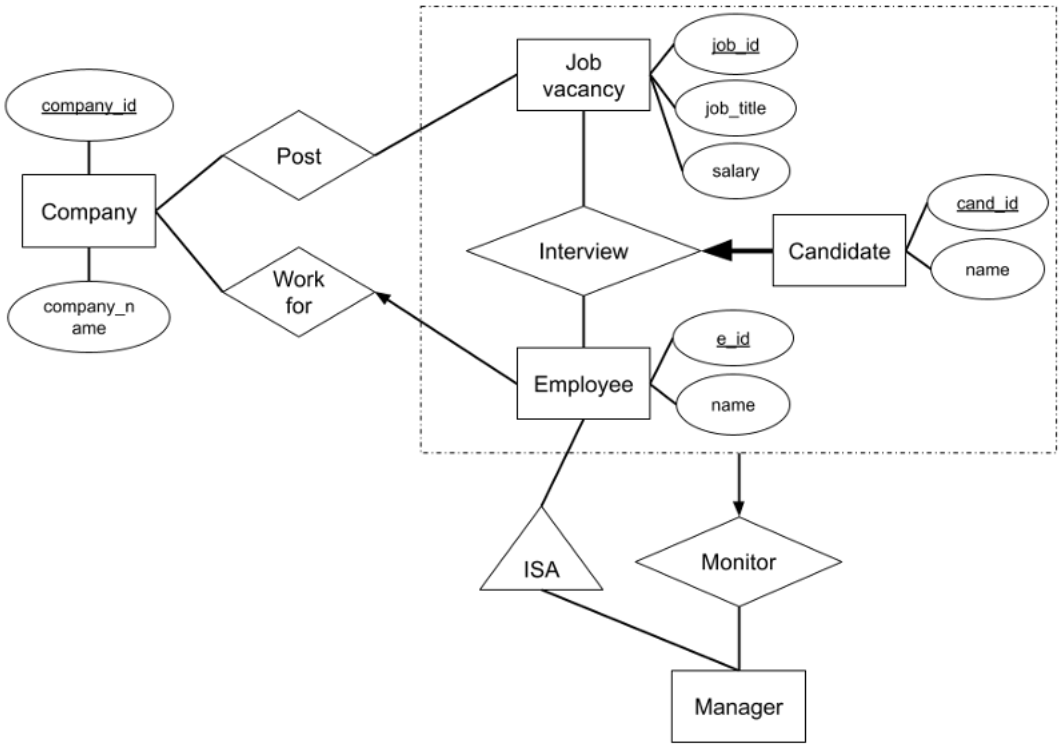
Your uniqname: _____

(Note: This particular sample contains problems on B+-trees and indexing, but that topic is **NOT** in scope for the midterm)

(page intentionally left blank. You can use it for rough work.)

Question 1: (15 points) ER diagrams

Given the diagram below, answer the following questions.



(a) Determine whether each of the following is true or false, given the constraints reflected by the above ER diagram for job application. No justification is required.

i. (1 point) A given employee may participate in multiple interviews of the same candidate for the same job vacancy.

ii. (1 point) The company can post multiple job vacancies for the same vacancy.

iii. (1 point) The manger has a key attribute that is identified in the diagram.

iv. (1 point) A candidate may interview for two different jobs vacancies at the same time.

v. (1 point) An interview may not be monitored by any manager.

(b) Assume Approach 2 for this question (i.e., fold tables when you can.)

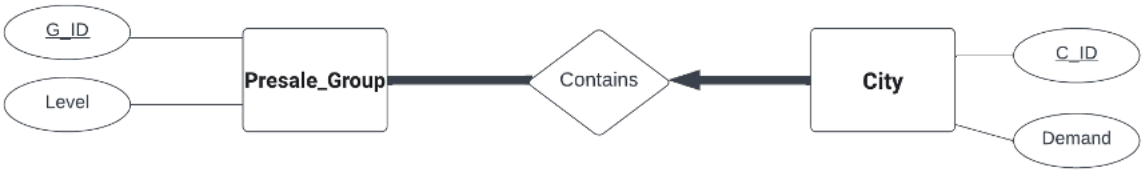
i. (3 points) How many tables will be required to represent the above diagram in the relational model, assuming you use Approach 2 when you can. (i.e., fold tables when you can). Assume that Manager has its own table that is distinct from that of an employee.

ii. (7 points) Write the CREATE TABLE statements for the table that contains the Candidate attributes. You may find the following information useful:

- Use type INTEGER for IDs and type VARCHAR2(20) for other attributes.
- The name of a candidate cannot be null.
- Assume that other tables exist with the attributes and tables named so that they are consistent with the ER diagram.

Question 2: (8 points) ER Diagram Design

Below is a possible ER Diagram representing how Ticketmaster stores information with regards to purchasing tickets for Taylor Swift's Midnights Tour or Beyoncé's Renaissance Tour. Some constraints are already shown in the figure.



On the answer book, fill in the missing constraints, attributes, and/or entities, given the information below. You can assume that all ID's can be used as unique identifiers for any given entity.

- The DB stores each fan's ID, credit card, and name.
- Ticketmaster requires that a ticket can not be purchased by more than one fan, but fans can purchase an unlimited amount of tickets.
- Ticketmaster also requires that for each ticket a fan purchases, they must also purchase an insurance plan that returns some money to the fan in case the fan cannot make it to the show. The DB will store the insurance plan's ID (globally unique) and coverage value. Each insurance plan is specific to one ticket.
- The DB will store the ticket's ID, level (VIP or Regular), and price.
- Finally, fans can optionally sign up for up to one presale group, which potentially allow them to purchase tickets before the general public. Ticketmaster will also store whether or not the fan is verified as eligible for the presale group they are signing up for.

Question 3: (23 points) Writing SQL queries

Consider the following schema for this and the next Question. Note: Primary keys [INTEGER] are underlined and attributes that are foreign keys are in bold.

Schema of Fakebook Database:

- `Users(user_id, first_name, last_name, current_city_id)`
- `Cities(city_id, city_name, state_name)`
- `Friends(requester_id, accepter_id)`

Notes:

- Each user will definitely have a `current_city_id`.
- We will not have (x, x) in `Friends`, i.e., a user is not friend with self. Also, if we have (x, y) in `Friends`, we are guaranteed to not have (y, x) in `Friends`.

Write SQL queries in Oracle syntax for the following questions. You can define additional views to build up towards the answer, if needed. There is no need to drop them. You can also use views defined in the problems below in subsequent parts of the question.

(a) (3 points) Zook, the founder of Fakebook, wants to gather more data on his users. He wants to know quickly what state his users are from. Create a view called `StateUsers(user_id, state_name)` that stores what state a user currently lives in.

(b) (5 points) Zook also wants to know how popular each user is on the platform. Create a view called `FriendCount(user_id, num_friends)` that stores the number of friends for each user. If a user does not have any friends, they should not appear in the view.

(c) (5 points) Zook wants to find for each state the users that have most friends. Write a SELECT query that returns a table with three columns: `(state_name, user_id, num_friends)`. The table contains, for each state, the `user_id`s in that state that have the highest number of friends, and the number of friends they have (irrespective of the state to which friends belong).

The resulting table should be sorted by `state_name` in ascending order first and `user_id` in descending order next (if multiple users from a state have the highest number of friends). Some example rows of an answer could be the following where users 23 and 24 are from Michigan and have 10 friends each, the highest for users in Michigan, and user 19 from Ohio has 8 friends, which is the highest for any user in that state. If a state only has users without friends, that state should not be included in the table.

State_Name	user_id	num_friends
MI	23	10
MI	24	10
OH	19	8

(d) (5 points) Create a view `Friends_of_Friends(f1, f2)` that contains the `user_id`s of a pair of users such that the two users are not directly friends, but they share a common friend. The view should contain each pair only once (e.g., both (1, 3) and (3, 1) should not be there).

(e) (5 points) Write a query that displays all (state_name, city_name) pairs for cities belonging to a state such that city_name is shared with another state's city name. The result should have two columns (state_name, shared_city_name).

Question 4: (12 points) Relational Algebra

This problem uses the same schema and assumptions as in the last problem. The schema is repeated here convenience:

- `Users(user_id, first_name, last_name, current_city_id)`
- `Cities(city_id, city_name, state_name)`
- `Friends(requester_id, accepter_id)`

Write a relational Algebra expression for each of the following.

(a) (5 points) Return the first name and last name of users who have zero friends (meaning the user has never had a friend request accepted and the user has never accepted a friend request).

(b) (2 points) Return the names of all cities whose names are in all states (these are city names that occur in all states).

(c) (5 points) Return the last names that are used with both of the first names 'Jim' and 'Pam'.

Question 5: (15 points) Functional Dependencies

Given a relation which contains that has the attributes (A, B, C, D, E, F, G) and the following functional dependencies

1. AD $\rightarrow$ CE
2. F $\rightarrow$ E
3. C $\rightarrow$ G
4. E $\rightarrow$ B
5. C $\rightarrow$ F

(a) (2 points) What are the key(s) (minimal) for the relation? List all keys if there are more than one.

(b) (4 points) Which, if any, of the above functional dependencies violate 3NF?

(c) Perform a BCNF decomposition on the first functional dependency that violates BCNF.

i. (2 points) Give the resulting relations after the above BCNF decomposition.

ii. (3 points) For the BCNF decomposition, give the dependencies that are preserved from the original list.

iii. (2 points) Give the keys for the two relations that result from the decomposition.

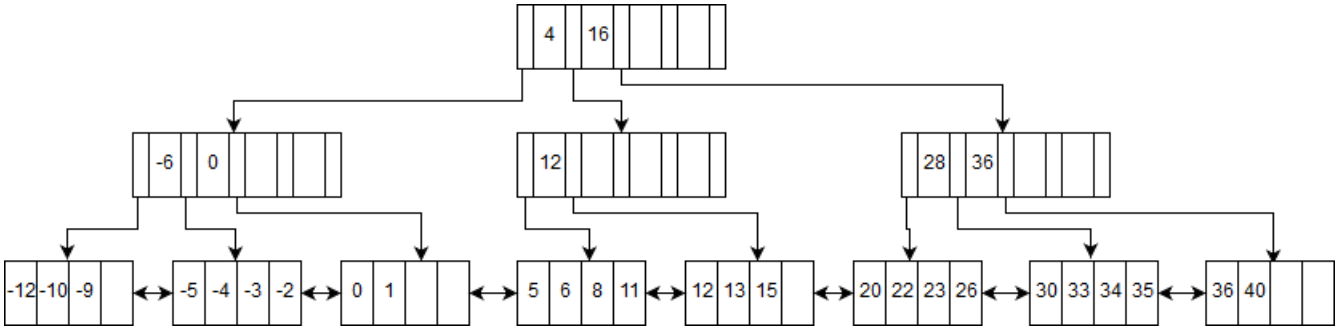
iv. (2 points) What is correct about the decomposition:

1. It satisfies lossless join property and is dependency-preserving.
2. It satisfies lossless join property, but is not dependency preserving.
3. It does not satisfy lossless join property, but it is dependency preserving.
4. It does not satisfy lossless join property and is not dependency preserving.

.

Question 6: (7 points) Tree Index

Consider the B+-tree index below on a table called `RatingTable` on attribute `rating` using Alternative 2. The values at the leaves are data entries for each rating in the table (* is not shown, but can be assumed to be there).



Note: Values in leaf nodes are data entries (i.e., k* values) – assume there is a * after each value.

The above B+ Tree index is used for the following query:

```
SELECT rating
FROM RatingTable
WHERE rating > 10;
```

Assume:

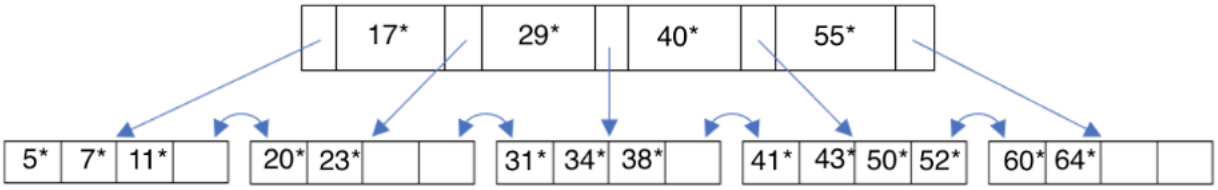
- The index is unclustered. Assume that fetching each data record from disk causes one page read.
- Each leaf node and inner node is on its own page
- None of the nodes are in memory. They are all on disk initially.

(a) (4 points) How many page reads is this query expected to require? Also, circle the nodes that are going to be read. Hint: Note that for this query, the rating values to answer the query can be found in leaf nodes in k* values and you should assume that the system takes advantage of that.

(b) (3 points) How many page reads would this query have required if all attributes (i.e., `SELECT *` instead of `SELECT rating` in the above query) that satisfy the two rating constraints in the WHERE clause were being fetched?

Question 7: (20 points) B+-tree inserts and deletes

Consider the B+-tree below of order 2 for both index and leaf nodes:



For inserts, assume no redistribution. Also assume that when splitting, the right node has equal or one more entry. For deletes, always attempt to redistribute first before considering a merge. If there is a choice in whom to borrow from, borrow from the left sibling.

(a) (10 points) Draw the final B+-tree after performing all of the following operations are done in the order shown:

1. Insert 36*
2. Insert 56*
3. Insert 65*
4. Insert 45*.

(b) (10 points) Start with the original B+-tree (i.e., no inserts done). Draw the final B+-tree after performing all the following operations in the order shown:

1. Delete 7*
2. Delete 23*
3. Delete 11*