1. Functional Dependencies (20 points)

Consider a relation $R(A, B, C, D, E)$ with functional dependencies

- $AB \rightarrow E$,
- $AE \rightarrow D$,
- $CDE \rightarrow AB$,
- $BD \rightarrow C$

(a) Enumerate all the keys in the above relation.
(b) Consider projection of $R$ into relations of three attributes; there are 10 of these. How many of these are in BCNF?

(c) How many of these 10 projected relations have exactly one superkey? How many of them have exactly three superkeys?
2. SQL (25 points)

Consider the tables `Frequents(drinker, bar)`, `Likes(drinker, beer)`, `Sells(bar, beer)`, with obvious meanings.
Write an SQL query to find all drinkers $d$ that frequent all bars that do not sell any beer liked by $d$. 
3. Datalog (20 points)

Write a stratified datalog program to find all drinkers $d$ that frequent all bars that do not sell any beer liked by $d$. 
4. **Indexes (25 points)**

The following questions are based on a grid file with two dimensions, $x$ and $y$. In each dimension, the space is divided into 10000 stripes by 9999 grid lines, not necessarily spaced evenly. An “index” in either dimension may be thought of as a sorted list of the values of the grid lines. On each block holding this list, we put the *number* (from 0 to 9999) of the stripe just below the lowest grid line on that block. Also needed in this structure is a bucket array $A$, which in C would be declared $A[10000][10000]$, whose elements are *pointers* to the block(s) holding the members of one bucket of the grid file. To lookup the point $(x, y)$, we search both dimensions’ indexes to find the stripes in each dimension where this point would lie, say stripe $i_x$ in the $x$ dimension and $i_y$ in the $y$ dimension. We then consult the array element $A[i_x][i_y]$ for a pointer to the bucket where $(x, y)$ may be found. Assume integers take 4 bytes, pointers take 8 bytes, and each block holds 1024 bytes. Also assume that the array $A$ is found in a known sequence of blocks, so that given $i_x$ and $i_y$ we may go directly to the block where $A[i_x][i_y]$ is found, without the need for another index.

(a) What is the minimum number of blocks needed to hold this structure (indexes and bucket array), not including any space needed to hold the buckets themselves?

(b) If we add whatever additional levels of index to the indexes for the dimensions that we wish, then what is the minimum number of disk I/O’s needed to lookup a point? Assume nothing is initially in memory and exclude any accesses to read the bucket itself.
5. **Query Optimization** (30 points)

Consider the following SQL query about relations \( R(A, B) \), \( S(B, C, D) \), and \( T(E, F) \).

```sql
select A, sum(D)
from R natural join S
where exists (select * from T where D > E)
  and exists (select * from T where A < F)
group by A;
```

Give the intuitively *best* query plan (represented as a tree) for the above query. Briefly, explain your reasoning. Note that in this problem we are not concerned with the specific methods used to implement the algebraic operators (join, selection, etc.)
6. **Transaction Processing (20 points)**

The next two questions refer to the following schedule:

\[ S = r_1(A); w_1(B); r_2(A); w_2(B); r_3(A); w_3(B); \]

That is, \( S \) is a serial schedule of three transactions, \( T_1, T_2, \) and \( T_3, \) each of which reads database element \( A \) and then writes database element \( B. \) Note that \( S \) is therefore “a serializable schedule equivalent to \( S. \)”

(a) How many schedules of the six actions in \( S \) are conflict-equivalent to \( S? \)

(b) How many schedules of the six actions in \( S \) are equivalent to \( S? \)

7. **2PL (10 points)**

Give a conflict-serializable schedule that cannot be produced by a 2 Phase Locking scheduler. Use as few transactions and read or write actions as possible.