Some questions have +I and –I for different options. The intent is that you sum over all options selected by a student, but in all cases, if the sum is less than 0, then make it 0.
1. (5 pts) Consider the B+ tree below.

Note that this tree does not show data nodes, and you do not need to see the data nodes to answer this question. At each leaf, N* is an index of the form <N, <page id, slot #>>, where N is the value of the search key.

Show the tree that results from inserting a records with search key 98 followed by 96 (do not use redistribution).
ANSWER for Question 1 (5 pts)

-2 if 96 and/or 98 does not appear at leaf

-2 for any tree that isn’t 4 levels. Use discretion on partial credit.
2. (5 pts) Consider the extendible hash table to the left. Assume Hash(x) = x. Show the result of inserting the following keys in order: 24 , 59

-1 for each misplaced key (order within a bin not important), and -1 for wrong local or global depth; grader – just need check that bins contain the correct content; don’t get hung up on checking pointers – mistakes there would probably be typos anyways

Answer here
3. Consider the relational schema \([A B C D]\) with FDs \(A \rightarrow B, B \rightarrow C, C \rightarrow A\)

(a) \((1 \text{ pt})\) Is the decomposition into \([B C]\) and \([A B D]\) dependency preserving? Explain.

1 pt for: No, \(C \rightarrow A\) cannot be assigned to any table/relation (full credit this time but no credit next time)
1 pt for: Yes, all FDs from an alternative minimal cover \((B \rightarrow C, C \rightarrow B, A \rightarrow B, B \rightarrow A)\) are assignable to each table

0 pts for: No or Yes with no explanation

(b) \((2 \text{ pts})\) Is each relation (i.e., of \([B C]\) and \([A B D]\) ) in BCNF? Explain.

\([B C]\) is in BCNF 1pt

\([A B D]\) is NOT in BCNF because \(A \rightarrow B\) is an FD of \([ABD]\) but \(A\) is not a key for \([ABD]\) 1pt

(c) \((1 \text{ pt})\) Is each relation in this alternative decomposition (i.e., of \([B C]\), \([A B]\), and \([A D]\) ) in BCNF? Explain.

Yes, \([BC]\) and \([AB]\) have only FDs with left hand sides that are keys for them; there are no FDs assignable to \([AD]\), so the BCNF condition that all FDs must have left-hand sides that are keys is trivially true
4. (3 pts) Consider the relation R with 7 attributes, $R = \{ A, B, C, D, E, F, G \}$. You are given the following functional dependencies: $Q = \{ AD \rightarrow BG, \ AE \rightarrow C, \ A \rightarrow F, \ AF \rightarrow D, \ EG \rightarrow BCF \}$. Give a minimal set of FDs equivalent to $Q$.

Can simplify LHS of $AD \rightarrow BG$ to $A \rightarrow BG$ (attribute closure of \{A\} includes D: \{A\} (A$\rightarrow$F) \{A,F\} (AF$\rightarrow$D) \{A,F,D\}…
Can simplify LHS of $AF \rightarrow D$ to $A \rightarrow D$ (attribute closure of \{A\} includes F: \{A\} (A$\rightarrow$F) \{A,F\}…

Can prune $AE \rightarrow C$ entirely, since attribute closure of \{A,E\} includes C anyways:

\{A,E\} (A$\rightarrow$BG) \{A, B, E, G\} (EG$\rightarrow$BCF or just consider EG$\rightarrow$C) \{A,B,C,…) …

In summary:

\begin{align*}
(AD \rightarrow BG) & \Rightarrow (A \rightarrow BG) \text{ since attribute closure of } A \text{ contains } D \\
(AF \rightarrow D) & \Rightarrow (A \rightarrow D) \text{ since } F \text{ in attribute closure of } A \\
(AE \rightarrow C) & \Rightarrow \text{prune, since attribute closure of } \{AE\} \text{ includes } C \text{ anyways:} \\
(A \rightarrow F) & \\
(EG \rightarrow BCF) & \text{minimal, or } (A \rightarrow BDFG, \ EG \rightarrow BCF) \text{ or} \\
(A \rightarrow B, A \rightarrow D, A \rightarrow F, A \rightarrow G, EG \rightarrow B, EG \rightarrow C, EG \rightarrow F)
\end{align*}
5. (3 pts) Circle all options that would correctly enforce a Complete Coverage constraint of Tab (in Tab1 and Tab2) in an SQL translation of the following UML fragment.

a) CREATE ASSERTION CompleteCoverageOfTab1AndTab2
   CHECK (SELECT COUNT (DISTINCT Tab.Tkey) FROM Tab)
   = (SELECT COUNT (DISTINCT Tab1.Tkey) FROM Tab1)
   + (SELECT COUNT(DISTINCT Tab2.Tkey) FROM Tab2)

b) CREATE ASSERTION CompleteCoverageOfTab1AndTab2
   CHECK (NOT EXISTS (SELECT Tab.Tkey FROM Tab
   EXCEPT SELECT Tab1.Tkey FROM Tab1
   EXCEPT SELECT Tab2.Tkey FROM Tab2))

1 points for each

1) CREATE ASSERTION CompleteCoverageOfTab1AndTab2
   CHECK (NOT EXISTS (SELECT Tab.Tkey FROM Tab
   EXCEPT SELECT Tab1.Tkey FROM Tab1
   EXCEPT SELECT Tab2.Tkey FROM Tab2))

c) CREATE ASSERTION CompleteCoverageOfTab1AndTab2
   CHECK (NOT EXISTS (SELECT Tab.Tkey FROM Tab
   EXCEPT (SELECT Tab1.Tkey FROM Tab1 UNION SELECT Tab2.Tkey FROM Tab2)))

d) CREATE ASSERTION CompleteCoverageOfTab1AndTab2
   CHECK (NOT EXISTS (SELECT Tab.Tkey FROM Tab
   WHERE Tab.Tkey NOT IN (SELECT Tab1.Tkey FROM Tab1) AND
   Tab.Tkey NOT IN (SELECT Tab2.Tkey FROM Tab2))

e) None of the above
   0 points for this

-1.5 for option (a) if with other options, and 0 total if by itself; if (e) is circled then it’s a 0 score, regardless of whether other options circled
6. (4 pts) Consider the UML fragment to the right and identify (circle) all equivalent table translations (i.e., those translations that faithfully enforce the constraints implied by the UML without regard to elegance) from those given below. You might receive partial credit for a brief explanation of your choices. UNIQUE(y) implies that y NOT NULL, but not vice versa. PK stands for PRIMARY KEY. FK stands for FOREIGN KEY.

(A) 4 points
CREATE TABLE X (x1, PK (x1))
CREATE TABLE R (x1, r1, z1, PK (x1, z1), FK (z1) refs Z, FK (x1) refs X)
CREATE TABLE Z (z1, PK (z1))
CREATE ASSERTION XparticipatesZ
CHECK (NOT EXISTS (SELECT * FROM X WHERE X.x1 NOT IN (SELECT R.x1 FROM R)))

-2 points

(B) CREATE TABLE X (x1, PK (x1))
CREATE TABLE R (x1, r1, z1, PK (x1, z1), FK (z1) refs Z, FK (x1) refs X)
CREATE TABLE Z (z1, PK (z1))
CREATE ASSERTION XparticipatesZ
CHECK (NOT EXISTS (SELECT * FROM X WHERE X.x1 NOT IN (SELECT R.x1 FROM R)))

-2 points

(C) CREATE TABLE X (x1, PK (x1))
CREATE TABLE R (x1, r1, z1, PK (x1, z1), FK (z1) refs Z, FK (x1) refs X)
CREATE TABLE Z (z1, PK (z1))
CREATE ASSERTION XparticipatesZ
CHECK (EXISTS (SELECT X.x1 FROM X) INTERSECT (SELECT R.x1 FROM R))

-1 points

(D) CREATE TABLE X (x1, PK (x1))
CREATE TABLE R (x1, r1, z1, PK (x1, z1), FK (z1) refs Z, FK (x1) refs X)
CREATE TABLE Z (z1, PK (z1))
CREATE ASSERTION XparticipatesZ
CHECK (NOT EXISTS (SELECT * FROM Z WHERE Z.z1 NOT IN (SELECT R.z1 FROM R)))

-2 points

(F) None of the others

0 points if circled, with or without other choices
7. (4 pts) Consider the following table definitions:

CREATE TABLE RelA (Aid integer, a1 integer, a2 integer, PRIMARY KEY (Aid))
CREATE TABLE RelB (Aid integer, Cid integer, b1 integer,
PRIMARY KEY (Aid, Cid, b1),
FOREIGN KEY (Aid) REFERENCES RelA,
FOREIGN KEY (Cid) REFERENCES RelC)
CREATE TABLE RelC (Cid integer, c1 integer, c2 integer, c3 integer,
PRIMARY KEY (Cid))

Circle all queries below that are equivalent to the query: SELECT C.c2, AVG (C.c3) AS avc3
FROM RelC C
WHERE C.c3 > 5
GROUP BY C.c2
HAVING COUNT(*) > 1

(a) SELECT C.c2, AVG (C.c3) AS avc3
FROM RelC C
WHERE C.c3 > 5 AND COUNT(*) > 1
GROUP BY C.c2
(b) SELECT C.c2, AVG (C.c3) AS avc3
FROM RelC C
WHERE C.c3 > 5
GROUP BY C.c2
HAVING 1 < (SELECT COUNT(*)
FROM RelC C2
WHERE C.c2 = C2.c2 AND C2.c3 > 5)
(c) SELECT C.c2, AVG (C.c3) AS avc3
FROM RelC C
GROUP BY C.c2
HAVING COUNT(*) > 1 AND C.c3 > 5
(d) SELECT Temp.c2, Temp.avc3
FROM (SELECT C.c2, AVG (C.c3) AS avc3, COUNT(*) AS c2count
FROM RelC C
WHERE C.c3 > 5
GROUP BY C.c2)
AS Temp
WHERE Temp.c2count > 1
(e) None of the above

1 pt
- 2pts
- 2pts
- 2pts
0 total
8. **(2 pts)** Consider the following table definitions:

CREATE TABLE Students ( 
    sid INTEGER, name CHAR(20), login CHAR(20), age INTEGER, gpa REAL, UNIQUE (name, age), PRIMARY KEY (sid) )

CREATE TABLE Enrolled ( 
    studid INTEGER, cid CHAR(20), grade CHAR(10), PRIMARY KEY (studid, cid), FOREIGN KEY (studid) REFERENCES Students (sid) )

Assume that Students contains the three rows shown below, and that Enrolled is empty.

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>53650</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53621</td>
<td>Smith</td>
<td>smith@ee</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>52134</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>2.9</td>
</tr>
</tbody>
</table>

Under the conditions above, circle all inserts below that will **FAIL**.

- **0.5 point**
  - a) INSERT INTO Students (sid, name, login, age, gpa) VALUES (53621, `Mike`, `mike@ee`, 17, 3.4)
- **0.5 point**
  - b) INSERT INTO Students (sid, name, login, age, gpa) VALUES (null, `Mike`, `mike@ee`, 17, 3.4)
- **0.5 point**
  - c) INSERT INTO Enrolled (studid, cid, grade) VALUES (51111, `EE101`, `B`)
- **0.5 point**
  - d) INSERT INTO Students (sid, name, login, age, gpa) VALUES (56781, `Jones`, `jw@cs`, 18, 2.5)

- e) None of the above will fail
9. (2 pts) Given a table `Book (Isbn, Title, Publisher, CopiesInStock)`, write an SQL statement that defines a VIEW called `STOCKER` that only enables access to fields `Isbn` and `CopiesInStock` of `Book`.

```
CREATE VIEW STOCKER (Isbn, Copies) AS
    SELECT B.Isbn, B.CopiesInStock FROM Book B
```

"(Isbn, Copies)" in SELECT clause is optional; 0 points if VIEW as SELECT isn’t evident.

10. (2 pts) You have two relations `R (C, F, G)` and `P (B, C, D, F, H)`. Using only relational algebra select (σ) and cross product (X) operators, express the natural join of `R` and `P`.

```
σ_{R.C=P.C AND R.F=P.F} (R X P) OR σ_{R.F=P.F} σ_{R.C=P.C} (R X P)
```

11. (1 pts) Consider the query `SELECT * FROM Apply, College
WHERE Apply.cName = College.cName AND Apply.major = 'CS' AND College.enrollment < 5000`

Which of the following indices could NOT be useful in speeding up this query’s execution?

(a) Tree-based index on `Apply.cName`
(b) Hash-based index on `Apply.major`
(c) Hash-based index on `College.enrollment`
(d) Hash-based index of `College.cName`
12. A colleague brings you three table definitions, summarized by these relational schema (R, S, T), with ‘a’ as a primary key for table T, ‘e’ the primary key for table S, and ‘a’ the primary key for table R. ‘e’ is a foreign key from R to S, and ‘a’ is a foreign key from R to T.

```
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>e</th>
<th>a</th>
<th>f</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

In addition to the table definitions, your colleague gives you this assertion, which is to be enforced in the DB.

```sql
CREATE ASSERTION GsPerB
CHECK (2 > ALL (SELECT COUNT (DISTINCT R.g)
FROM T, R
WHERE T.a = R.a
GROUP BY T.b)))
```

(a) (1 pt) Is the functional dependency \( a \rightarrow b \) implied by any of three table definitions and the assertion? Explain.

Yes, because \( a \) is a key for \([a b]\) (or Table T)

(a) (1 pt) Is the functional dependency \( b \rightarrow g \) implied by any of the three table definitions and the assertion? Explain.

yes, \( b \rightarrow g \) is enforced by the assertion

**ADDITIONAL DISCUSSION (NOT GRADED)**

From tables: \( a \rightarrow b \); \( c \rightarrow c, d \); \( a \rightarrow e, f, g \) (could combine first and last); \( b \rightarrow g \) (from assertion);

but \( a \rightarrow g \) is redundant (because \( a \rightarrow b \) and \( b \rightarrow g \)), and is eliminated, so minimal set is

\( a \rightarrow b, e, f \); \( c \rightarrow c, d \); \( b \rightarrow g \)

Given you want to enforce these functional dependencies, are the three tables that your colleague gave you dependency preserving? Not Dependency Preserving, since \( b \rightarrow g \) not assignable to any table (which is why it had to be enforced through assertion – NOT good in this case, because ANOTHER decomposition would have NOT needed assertion)