Assessment was due by Sat, 2023-12-09 11:30:00

COMP 421 Final Exam Fall 2023

Note: There are a total of 110 points on this exams. The highest score you can get is a 100. Essentially there are 10 points that you can miss and still get a 100%.

Don't panic!

You have 180 minutes to finish the exam.

- You *must* stay in full screen mode. Points removed for leaving full screen mode
- You must hand this final exam in on time.
    - Points removed for late submissions.
    - Only your **first** submission will be accepted.
    - Avoid accidental submissions. Fill in your name when you are ready to submit.
- Points removed for accidental submissions.

I recommend that you have several pieces of scrap paper to doodle notes on during the exam.

Consider this final **closed book**.
You MAY use your hand written notes. They MUST be on paper as you may not switch screens after starting the exam.

You **MAY NOT** Google or use other external websites for **answers** or copy from a friend. Do not paste information into your exam unless it was copied from your exam. You **MAY NOT** receive help from anyone.

If you do not know the origin of material you should not paste it into this exam. All material pasted into this exam must originate from this exam. This

implies, but is not limited to, copying from previous assignments, copying from text messages, or copying from **any** website.

You **MUST** use the Google Chrome browser.

The browser will change input box color green to indicate correctness. A black or red box indicates an incorrect answer.

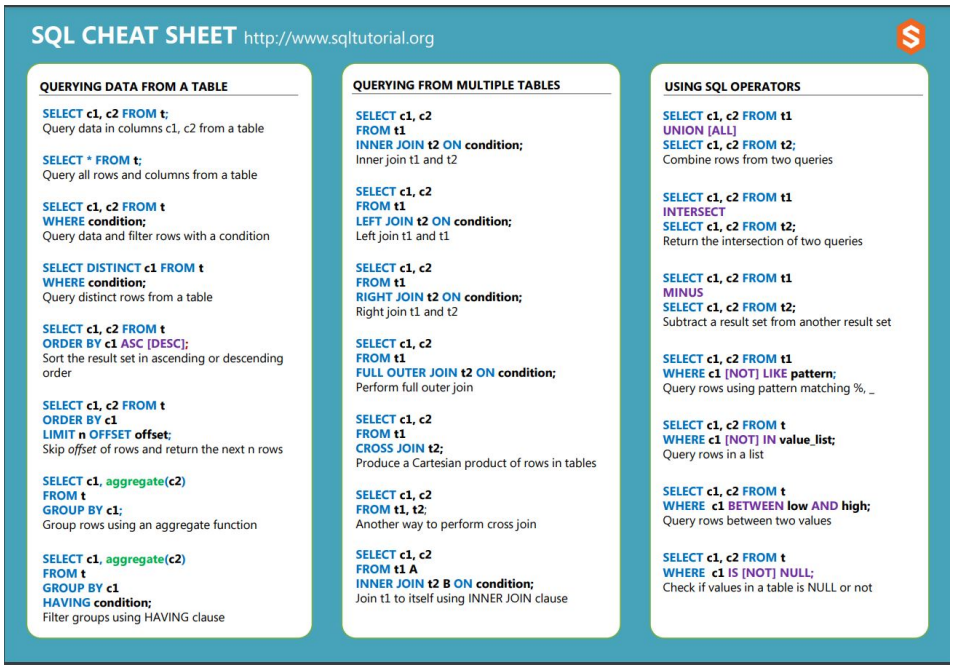
Note that HTML select statements with drop-downs are simple multiple choice questions. No highlighting of correct answers are done for select questions.

Green highlight should just assist you. If you believe your answer is correct and the input box did not turn green, continue on. Per the syllabus, highlighting is simply an aide not a guarantee.

Note: For database queries that are applied to two databases, **two** green lights are required to get any credit for the question.

SQL Tutorial Cheat Sheet

Following are three SQL tutorial cheat sheets available from http://www.sqltutorial.org



SQL CHEAT SHEET http://www.sqltutorial.org

QUERYING DATA FROM A TABLE

```
SELECT c1, c2 FROM t;
```
Query data in columns c1, c2 from a table

```
SELECT * FROM t;
```
Query all rows and columns from a table

```
SELECT c1, c2 FROM t
WHERE condition;
```
Query data and filter rows with a condition

```
SELECT DISTINCT c1 FROM t
WHERE condition;
```
Query distinct rows from a table

```
SELECT c1, c2 FROM t
ORDER BY c1 ASC [DESC];
```
Sort the result set in ascending or descending order

```
SELECT c1, c2 FROM t
ORDER BY c1
LIMIT n OFFSET offset;
```
Skip *offset* of rows and return the next n rows

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1;
```
Group rows using an aggregate function

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
HAVING condition;
```
Filter groups using HAVING clause

QUERYING FROM MULTIPLE TABLES

```
SELECT c1, c2
FROM t1
INNER JOIN t2 ON condition;
```
Inner join t1 and t2

```
SELECT c1, c2
FROM t1
LEFT JOIN t2 ON condition;
```
Left join t1 and t1

```
SELECT c1, c2
FROM t1
RIGHT JOIN t2 ON condition;
```
Right join t1 and t2

```
SELECT c1, c2
FROM t1
FULL OUTER JOIN t2 ON condition;
```
Perform full outer join

```
SELECT c1, c2
FROM t1
CROSS JOIN t2;
```
Produce a Cartesian product of rows in tables

```
SELECT c1, c2
FROM t1, t2;
```
Another way to perform cross join

```
SELECT c1, c2
FROM t1 A
INNER JOIN t2 B ON condition;
```
Join t1 to itself using INNER JOIN clause

USING SQL OPERATORS

```
SELECT c1, c2 FROM t1
UNION [ALL]
SELECT c1, c2 FROM t2;
```
Combine rows from two queries

```
SELECT c1, c2 FROM t1
INTERSECT
SELECT c1, c2 FROM t2;
```
Return the intersection of two queries

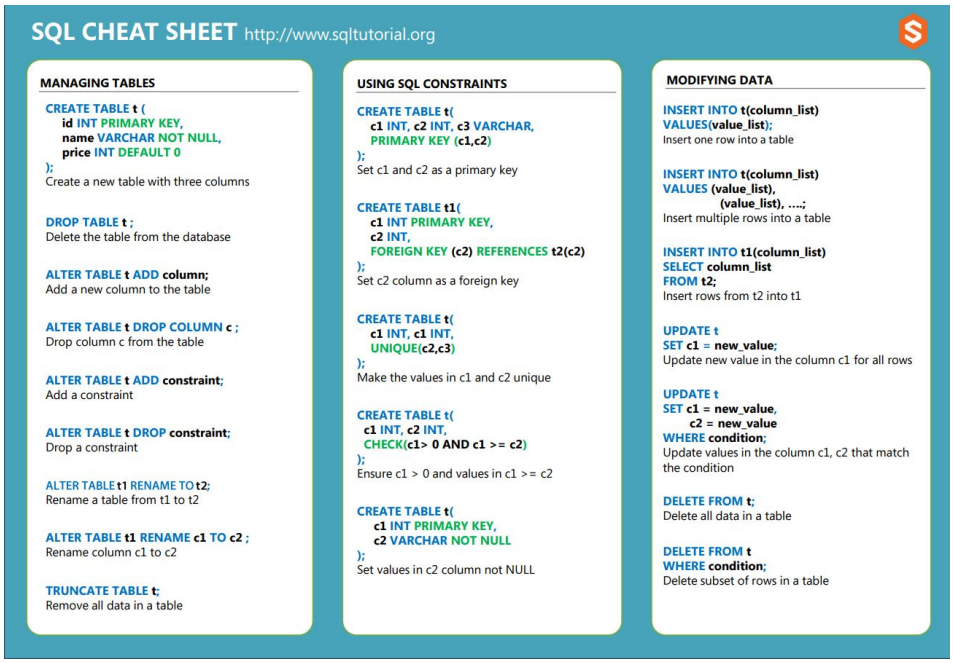
```
SELECT c1, c2 FROM t1
MINUS
SELECT c1, c2 FROM t2;
```
Subtract a result set from another result set

```
SELECT c1, c2 FROM t1
WHERE c1 [NOT] LIKE pattern;
```
Query rows using pattern matching %, _

```
SELECT c1, c2 FROM t
WHERE c1 [NOT] IN value_list;
```
Query rows in a list

```
SELECT c1, c2 FROM t
WHERE  c1 BETWEEN low AND high;
```
Query rows between two values

```
SELECT c1, c2 FROM t
WHERE  c1 IS [NOT] NULL;
```
Check if values in a table is NULL or not



SQL CHEAT SHEET http://www.sqltutorial.org

MANAGING TABLES

```
CREATE TABLE t (
   id INT PRIMARY KEY,
   name VARCHAR NOT NULL,
   price INT DEFAULT 0
);
```
Create a new table with three columns

```
DROP TABLE t ;
```
Delete the table from the database

```
ALTER TABLE t ADD column;
```
Add a new column to the table

```
ALTER TABLE t DROP COLUMN c ;
```
Drop column c from the table

```
ALTER TABLE t ADD constraint;
```
Add a constraint

```
ALTER TABLE t DROP constraint;
```
Drop a constraint

```
ALTER TABLE t1 RENAME TO t2;
```
Rename a table from t1 to t2

```
ALTER TABLE t1 RENAME c1 TO c2 ;
```
Rename column c1 to c2

```
TRUNCATE TABLE t;
```
Remove all data in a table

USING SQL CONSTRAINTS

```
CREATE TABLE t(
   c1 INT, c2 INT, c3 VARCHAR,
   PRIMARY KEY (c1,c2)
);
```
Set c1 and c2 as a primary key

```
CREATE TABLE t1(
   c1 INT PRIMARY KEY,
   c2 INT,
   FOREIGN KEY (c2) REFERENCES t2(c2)
);
```
Set c2 column as a foreign key

```
CREATE TABLE t(
   c1 INT, c1 INT,
   UNIQUE(c2,c3)
);
```
Make the values in c1 and c2 unique

```
CREATE TABLE t(
  c1 INT, c2 INT,
  CHECK(c1> 0 AND c1 >= c2)
);
```
Ensure c1 > 0 and values in c1 >= c2

```
CREATE TABLE t(
   c1 INT PRIMARY KEY,
   c2 VARCHAR NOT NULL
);
```
Set values in c2 column not NULL

MODIFYING DATA

```
INSERT INTO t(column_list)
VALUES(value_list);
```
Insert one row into a table

```
INSERT INTO t(column_list)
VALUES (value_list),
       (value_list), ...;
```
Insert multiple rows into a table

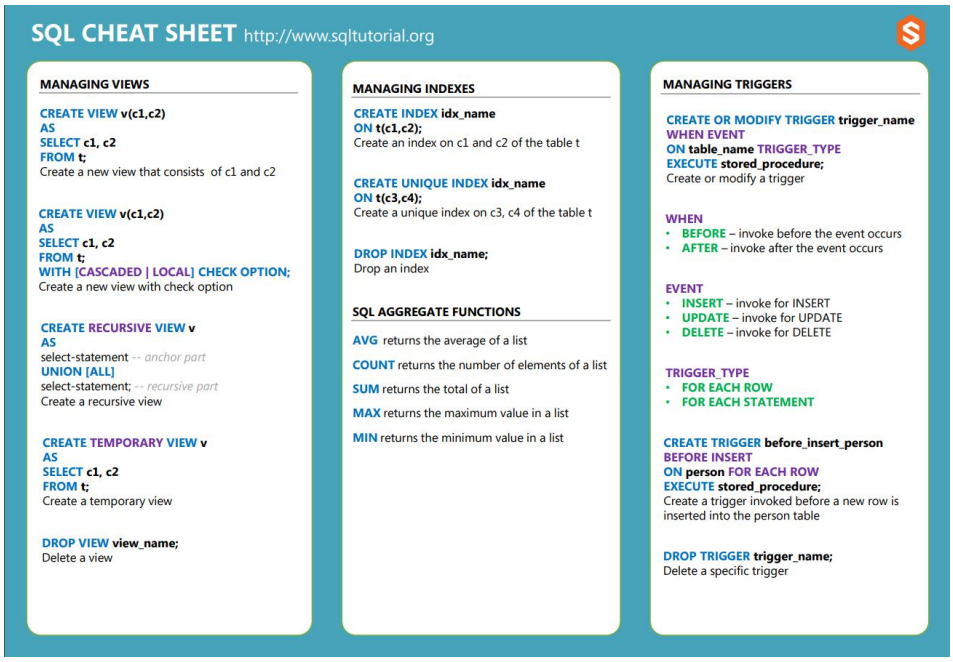
```
INSERT INTO t1(column_list)
SELECT column_list
FROM t2;
```
Insert rows from t2 into t1

```
UPDATE t
SET c1 = new_value;
```
Update new value in the column c1 for all rows

```
UPDATE t
SET c1 = new_value,
    c2 = new_value
WHERE condition;
```
Update values in the column c1, c2 that match the condition

```
DELETE FROM t;
```
Delete all data in a table

```
DELETE FROM t
WHERE condition;
```
Delete subset of rows in a table



Database Schema

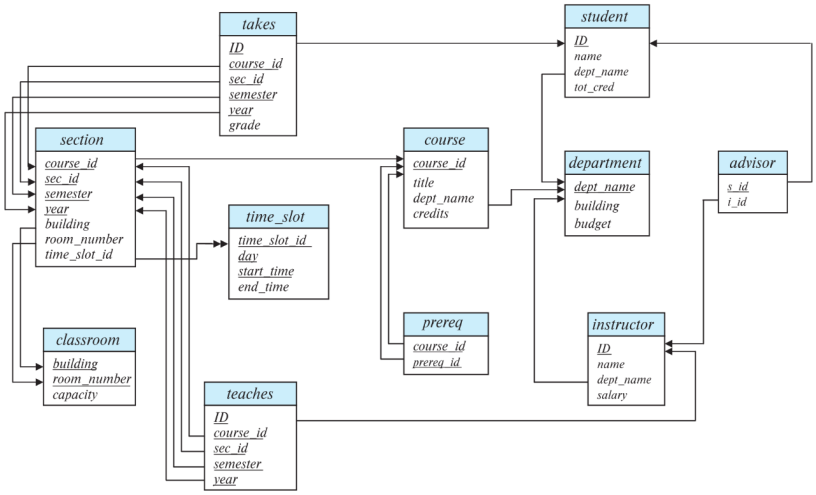
Here are the tables you'll find for the database used in the midterm. Your queries will be run against two versions of the database. One of the databases will be much smaller and only contain a subset of the information.

```
CREATE TABLE classroom
     (building     varchar(15),
     room_number  varchar(7),
     capacity     numeric(4,0),
     primary key (building, room_number)
     )
CREATE TABLE department
     (dept_name   varchar(20),
     building    varchar(15),
     budget      numeric(12,2) check (budget > 0),
     primary key (dept_name)
     )
CREATE TABLE course
     (course_id   varchar(8),
     title       varchar(50),
     dept_name   varchar(20),
     credits     numeric(2,0) check (credits > 0),
     primary key (course_id),
     foreign key (dept_name) references department (
```

```
    on delete set null
    )
CREATE INDEX idx_course_dept_name ON Course(dept_nam
CREATE TABLE instructor
    (ID          varchar(5),  -- instructor's ID
    name        varchar(20) not null,
    dept_name   varchar(20),
    salary      numeric(8,2) check (salary > 29000)
    primary key (ID),
    foreign key (dept_name) references department (
    on delete set null
    )
CREATE INDEX idx_instructor_id ON Instructor(ID)
CREATE INDEX idx_instructor_dept_name ON Instructor(
CREATE TABLE section
    (course_id    varchar(8),
    sec_id       varchar(8),
    semester     varchar(6)
    check (semester in ('Fall', 'Winter', 'Spring',
    year         numeric(4,0) check (year > 1701 an
    building     varchar(15),
    room_number  varchar(7),
    time_slot_id varchar(4),
    primary key (course_id, sec_id, semester, year)
    foreign key (course_id) references course (cour
    on delete cascade,
    foreign key (building, room_number) references
    on delete set null
    )
CREATE INDEX idx_section_year ON Section(year)
CREATE INDEX idx_section_semester ON Section(semeste
CREATE TABLE teaches
    (ID         varchar(5),  -- instructor's ID
    course_id  varchar(8),
    sec_id     varchar(8),
    semester   varchar(6),
    year       numeric(4,0),
    primary key (ID, course_id, sec_id, semester, y
    foreign key (course_id, sec_id, semester, year)
    references section (course_id, sec_id, semester
```

```
     on delete cascade,
     foreign key (ID) references instructor (ID)
     on delete cascade
     )
CREATE INDEX idx_teaches_id ON Teaches(ID)
CREATE INDEX idx_teaches_year ON Teaches(year)
CREATE INDEX idx_teaches_semester ON Teaches(semeste
CREATE TABLE student
     (ID         varchar(5),
     name       varchar(20) not null,
     dept_name  varchar(20),
     tot_cred   numeric(3,0) check (tot_cred >= 0),
     primary key (ID),
     foreign key (dept_name) references department (
     on delete set null
     )
CREATE UNIQUE INDEX idx_student_id ON Student(ID)
CREATE INDEX idx_student_dept_name ON Student(dept_n
CREATE TABLE takes
     (ID         varchar(5), -- Student ID
     course_id  varchar(8),
     sec_id     varchar(8),
     semester   varchar(6),
     year       numeric(4,0),
     grade      varchar(2),
     primary key (ID, course_id, sec_id, semester, y
     foreign key (course_id, sec_id, semester, year)
     references section (course_id, sec_id, semester
     on delete cascade,
     foreign key (ID) references student (ID)
     on delete cascade
     )
CREATE INDEX idx_takes_id ON Takes(ID)
CREATE INDEX idx_takes_semester ON Takes(semester)
CREATE INDEX idx_takes_year ON Takes(year)
CREATE TABLE advisor
     (s_ID       varchar(5),
     i_ID       varchar(5),
     primary key (s_ID),
     foreign key (i_ID) references instructor (ID)
```

```
      on delete set null,
      foreign key (s_ID) references student (ID)
      on delete cascade
      )
CREATE INDEX idx_advisor_instructor_id ON Advisor(i_
CREATE TABLE time_slot
      (time_slot_id   varchar(4),
      day             varchar(1),
      start_hr        numeric(2) check (start_hr >= 0
      start_min       numeric(2) check (start_min >= 0
      end_hr          numeric(2) check (end_hr >= 0 an
      end_min         numeric(2) check (end_min >= 0 a
      primary key (time_slot_id, day, start_hr, start
      )
CREATE TABLE prereq
      (course_id      varchar(8),
      prereq_id       varchar(8),
      primary key (course_id, prereq_id),
      foreign key (course_id) references course (cour
      on delete cascade,
      foreign key (prereq_id) references course (cour
      )
```



Scratch area

The following scratch space can be used to help develop and test queries against a database described above. The database used by the exam grader will be different.

Execute Minimize Output

Questions For a total of 110 points

SQL Queries 60 points

In this section, you will write SQL queries for the university described in the book. Your queries will be tested immediately against two different databases. If your queries output matches the expected output, the displayed answers will be outlined in green. Your actual score will be determined when your query is tested against a different database but green feedback should mean that you are on track to receive full credit.

List.Instructors.1: List the names of the instructors in the department of Accounting.
List the names in reverse alphabetical order.

Execute 5 points Minimize Output

List.Students.And.Instructors.1: List the names of people in the department of Accounting. This should be a single column table with the names of both the instructors and the students in the department of Accounting.
List the names in reverse alphabetical order.

Execute 7.5 points Minimize Output

List.Instructor.1.3: List the names of the instructors in the department of Accounting who have a name with exactly 3 letters in their name.
List the names in alphabetical order.

Execute 5 points Minimize Output

List.Instructor.Did.Not.Teach.All.1: List the names of the instructors in the department of Accounting who have **never** taught at least one Accounting course.
Another way to say this is: list all the names of Accounting instructors except those who taught all of the Accounting courses.
For example, if the Accounting department offers twelve courses, your answer should include any Accounting instructor's name who hasn't taught at least one section of every offered Accounting course.

Execute 7.5 points Minimize Output

Update.Credits.1: Update the credits for Accounting courses such that they are worth 50% more than before.
NOTE$_1$: With all SQL statements like this that modify the database, you either need to refresh the web page before rerunning or handle the fact that the statement was previously run.
NOTE$_2$: The grader will run your modifications against both databases which should report **two** empty green boxes. Then the grader will run two select statments against both databases to verify the modification which should be in **four** more green boxes. A correct answer should have a total of **six** green boxes

Execute 7.5 points Minimize Output

Relational.Algebra.1: Create a SQL query for the relational expression:

$$\Pi_{\text{title}}(\sigma_{\text{semester}='Fall' \wedge \text{dept\_name}='Accounting'}(\rho_s(\text{Section}) \bowtie_{\text{c.course\_id}=\text{s.course\_id}} \rho_c(\text{Course})))$$
$$-$$
$$\Pi_{\text{title}}(\sigma_{\text{semester}='Winter' \wedge \text{dept\_name}='Accounting'}(\rho_s(\text{Section}) \bowtie_{\text{c.course\_id}=\text{s.course\_id}} \rho_c(\text{Course}))$$
$$-$$
$$\Pi_{\text{title}}(\sigma_{\text{semester}='Spring' \wedge \text{dept\_name}='Accounting'}(\rho_s(\text{Section}) \bowtie_{\text{c.course\_id}=\text{s.course\_id}} \rho_c(\text{Course}))$$
$$-$$
$$\Pi_{\text{title}}(\sigma_{\text{semester}='Summer' \wedge \text{dept\_name}='Accounting'}(\rho_s(\text{Section}) \bowtie_{\text{c.course\_id}=\text{s.course\_id}} \rho_c(\text{Course}))$$

Execute 5 points Minimize Output

Largest.Classroom.1: List the building and room_number of the classroom(s) of a Accounting class section that has the largest capacity.

Execute 7.5 points Minimize Output

List.Department.Budget.Maximum.Salary.1: List all the department names, their budgets, and the maximum instructor salary for each department. List the departments by largest budget to smallest budget.
NOTE: You can be assured that all departments have at least one instructor.

Execute 5 points Minimize Output

List.Students.With.Their.Advisor.1: List the name of each student in the Accounting department who has an advisor. In addition to the name of the

student, give the name of the student's advisor.
Note: Only the student has to be in the Accounting department.

Execute 5 points Minimize Output

List.Students.And.Advisor.If.Exists.1: List the name of each student in the Accounting department and if they have an advisor list the advisor name. If the student doesn't have an advisor leave the advisor name NULL.

Execute 5 points Minimize Output

Query processing 20 points

In this section, you will show your knowledge of Chapter 15 query processing

Calculator: You may use this box as a calculator. Just type in any expression that can be evaluated by a JavaScript eval. For example, `2 ** 0.5` will show you the result of the square root of 2 in the red box to the right. Leaving the box empty or filled in will **NOT** affect your grade.

Enter an expression undefined

Let relation r_1 have 20,000 tuples where each block holds 25 tuples. Let relation r_2 have 45,000 tuples where each block holds 30 tuples.

Nested.Loop.R1.Outer.1: How many block transfers are required if a nested-loop join is done with r_1 as the outer loop and there is only one buffer for r_1 and one buffer for r_2?
NOTE: As previously done in class, do not count the number of block transfers required for output of the nested-loop join.

Enter an expression 2 points

Nested.Loop.R2.Outer.1: How many block transfers are required if a nested-loop join is done with r_2 as the outer loop and there is only one buffer for r_1 and one buffer for r_2?
NOTE: As previously done in class, do not count the number of block transfers required for output of the nested-loop join.

Enter an expression 2 points

Nested.Loop.Which.Outer.1: In order to minimized the number of block transfers, which relation should be used for the outer loop?

1 point

Block.Nested.Loop.R1.Outer.1: How many block transfers are required if a block nested-loop join is used with r_1 as the outer relation?
Assume the enhanced version of block nested-loop join where M blocks are used for buffering of input of r_1 and r_2 and one buffer reserved to collect the output of the join.
NOTE$_1$: As previously done in class, do not count the number of block transfers required for output of the block nested-loop join.
NOTE$_2$: Remember, the evaluation box allows you to use functions floor(), ceil(), and log().

Enter an expression 5 points

Sort.Merge.R1.1: Assuming there are 25 blocks of memory available for sorting r_1, complete the following table below. For each pass, how many runs exists in that pass and the number of blocks in the largest run.

NOTE$_1$: The table has notation for up to four passes. If the merge-sort completes in less than four passes, those *extra* passes would note that one run exists and that the largest run is the number of blocks in r_1.
NOTE$_2$: Remember, the evaluation box allows you to use functions floor(), ceil(), and log().

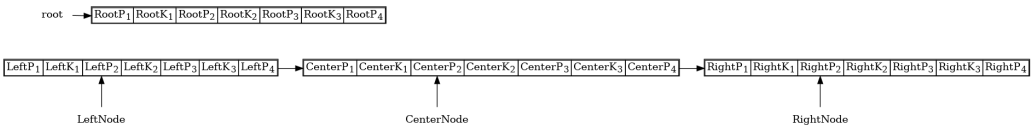
Pass number	Number runs that exist	Number of blocks in the largest run
Pass 1		
Pass 2		
Pass 3		
Pass 4		

10 points

B-plus trees 30 points

In this section, you will show your knowledge of B$^+$ trees.

Assume the following B$^+$ tree with $n = 4$ and the **Takes** relation and a search key of grade:



Takes Relation

sec_id	semester	ID	course_id	year	grade
1	Fall	Student1	COMP110	2001	A+
2	Fall	Student6	COMP210	2020	F
3	Spring	Student2	COMP116	2006	C
4	Summer	Student3	COMP311	2019	D
5	Summer	Student5	COMP126	2012	B
6	Winter	Student4	COMP283	2023	B+
7	Winter	Student6	COMP311	2021	A
8	Spring	Student7	COMP421	2022	C+
9	Fall	Student8	COMP550	2021	A

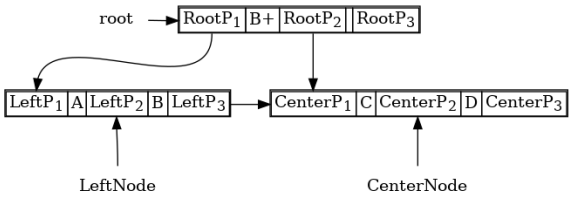
Complete.B.Plus.Tree.1: Complete the indices for relation Takes using a search key of grade. You should assume the same notation used in class and the same algorithm used in the book.
To facilitate having only one single correct answer, field CenterK3 must be 'C+'. It may help you to start building your tree from field CenterK3.
NOTE: Ascending alphanumeric sort order for grades is A, A+, A-, B, B+, B-, C, C+, C-, D, D+, D-, F

Field	Value
RootP1	

RootK1	
RootP2	
RootK2	
RootP3	
RootK3	
RootP4	
LeftP1	
LeftK1	
LeftP2	
LeftK2	
LeftP3	
LeftK3	
LeftP4	
CenterP1	
CenterK1	
CenterP2	
CenterK2	
CenterP3	
CenterK3	
CenterP4	
RightP1	
RightK1	
RightP2	
RightK2	
RightP3	
RightK3	
RightP4	

15 points

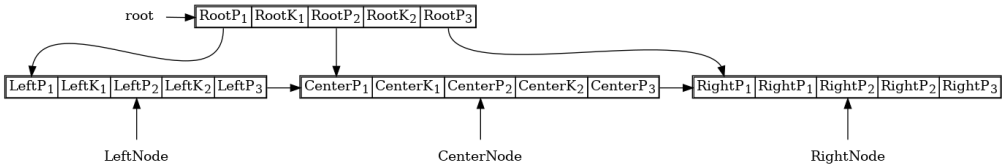
Assume the following B^+ tree with $n = 3$ and the **Takes** relation and a search key of grade:



Takes Relation

sec_id	semester	ID	course_id	year	grade
3	Spring	Student2	COMP116	2006	C
4	Summer	Student3	COMP311	2019	D
5	Summer	Student5	COMP126	2012	B
7	Winter	Student6	COMP311	2021	A

Insert.B.Plus.Tree.1: Complete the following table of grade search keys and pointers after an insert of record '8 Spring Student7 COMP421 2022 C+'.
Note: You can be assured that RootP1 points to LeftNode, RootP2 points to CenterNode, and RootP3 points to RightNode. **NOTE:** Ascending alphanumeric sort order for grades is A, A+, A-, B, B+, B-, C, C+, C-, D, D+, D-, F



Takes Relation

sec_id	semester	ID	course_id	year	grade
3	Spring	Student2	COMP116	2006	C
4	Summer	Student3	COMP311	2019	D
5	Summer	Student5	COMP126	2012	B
7	Winter	Student6	COMP311	2021	A
8	Spring	Student7	COMP421	2022	C+

Field	Value
RootP1	
RootK1	
RootP2	
RootK2	
RootP3	
LeftP1	
LeftK1	
LeftP2	
LeftK2	
LeftP3	
CenterP1	
CenterK1	
CenterP2	

CenterK2	
CenterP3	
RightP1	
RightK1	
RightP2	
RightK2	
RightP3	

15 points
