Assessment was due by Tue, 2023-05-09 11:15:00

Final Exam COMP 421 Spring 2023

Note: There are a total of 115 points on this exam. The highest grade you can get is 100 points. Think of this as partial credit, as 15 points whose wording isn't quite perfect, or as a buffer for questions you didn't study for.

Given the extra points, manage your time wisely!

Disclaimer: All data in the first database is taken from LegiScan website. I have made some minor modifications to limit the possibility of hard coding answers. For example, Jesse Helms, a deceased NC legislator, and James K. Polk 11th President of the United States and UNC alumnus, have been added to one of the databases used.

None of these changes are made to present a political bias. Any errors in the transcription of the data from LegiScan to the exam database are mine. If you see any errors, please let me know so that I may make timely corrections.

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.

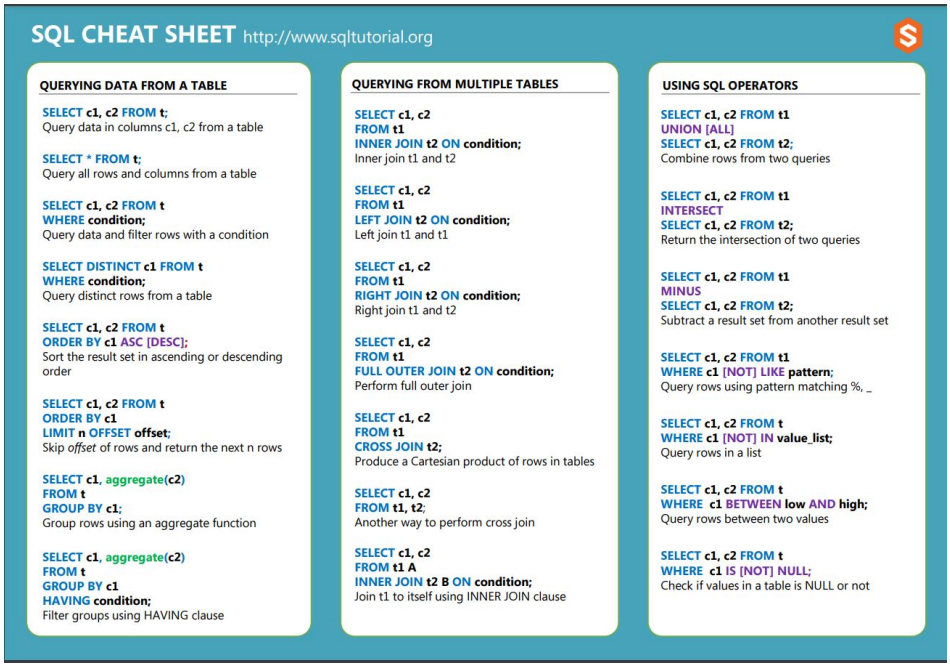
Note that HTML select statements with drop-downs are simple multiple choice questions. No highlighting of correct answers are done for select questions.

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

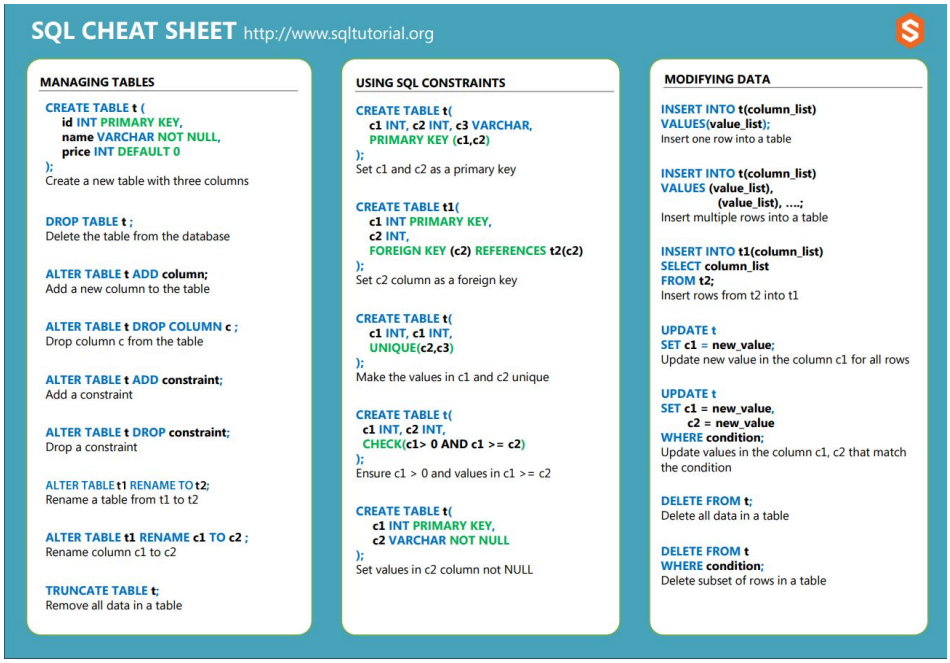
```
SELECT c1, c2 FROM t1
MINUS
SELECT c1, c2 FROM t2;
```
Subtract a result set from another result set

```
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

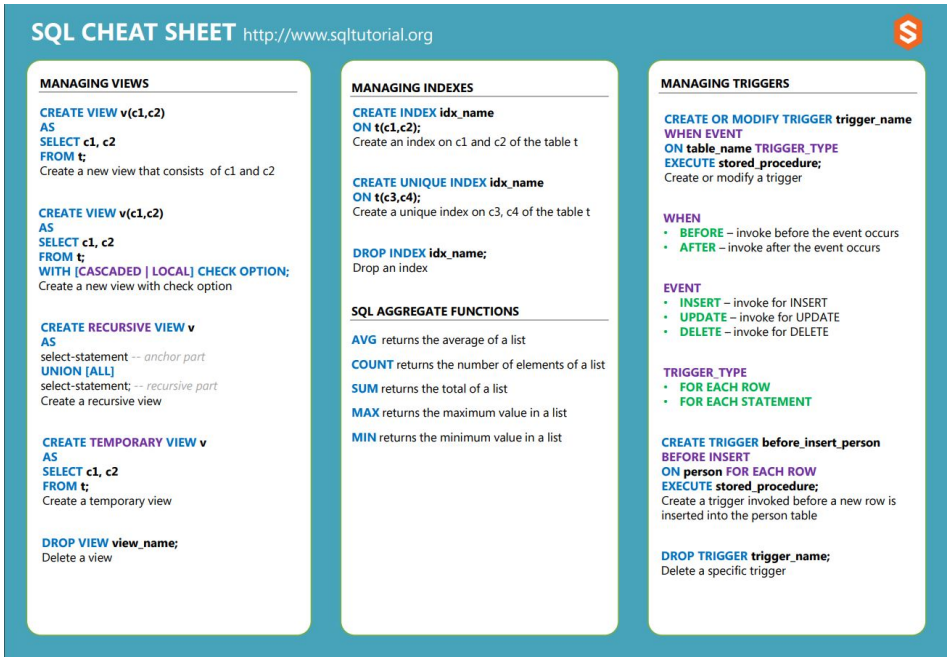
```
INSERT INTO t1(column_list)
SELECT column_list
FROM t2;
```
Insert rows from t2 into t1

```
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



Questions For a total of 115 points

Legislature Database Schema

Here are the tables you'll find for the database used in the final exam. 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 Legislators
     (legislator_id INTEGER PRIMARY KEY,
     party TEXT CHECK(party IN (`Democratic`, `Republicar
     role text CHECK(role IN (`Senator`, `Representative`
     name text,
     first_name text,
     middle_name text,
     last_name text,
     district text CHECK(SUBSTR(district, 1,3) IN (`HD-`,
     )
CREATE TABLE Bills
     (bill_id INTEGER PRIMARY KEY,
     title TEXT NOT NULL,
     status TEXT CHECK(status IN (`Introduced`, `Passes`,
     status_date TEXT, -- YYYY-MM-DD
     url TEXT)
```

```
CREATE TABLE Roll_calls
     (roll_call_id INTEGER PRIMARY KEY,
     bill_id INTEGER REFERENCES Bills(bill_id) NOT NULL,
     description TEXT,
     date TEXT, -- YYYY-MM-DD if available
     yea INTEGER UNSIGNED, -- Count of Yeas
     nay INTEGER UNSIGNED, -- Count of Nays
     not_voting INTEGER UNSIGNED, -- Count of not voting
     absent INTEGER UNSIGNED, -- Count of absent
     passed INTEGER CHECK(passed IN (0, 1)),
     status_date TEXT, -- YYYY-MM-DD
     chamber TEXT CHECK(chamber IN (`House`, `Senate`)))
CREATE TABLE Votes
     (roll_call_id INTEGER REFERENCES Roll_calls(Roll_cal
     legislator_id INTEGER REFERENCES Legislators(legisla
     vote_text TEXT CHECK(vote_text IN (`Yea`, `Nay`, `Ab
     UNIQUE(roll_call_id, legislator_id))
CREATE TABLE Sponsors
     (bill_id INTEGER REFERENCES Bills(bill_id),
     legislator_id INTEGER REFERENCES Legislators(legisla
     UNIQUE(bill_id, legislator_id))
CREATE TABLE Subject_names
     (subject_id INTEGER PRIMARY KEY,
     subject_name TEXT,
     UNIQUE(subject_name))
CREATE TABLE Bill_subjects
     (subject_id INTEGER REFERENCES Subject_names(subject
     bill_id INTEGER REFERENCES Bills(bill_id) NOT NULL,
     UNIQUE(subject_id, bill_id))
```

Legislator Scratch area

The following scratch space can be used to help develop and test queries against the **first** of the two legislator databases described above.

```
WITH Sponsor_counts AS (SELECT COUNT(*) as bill_count, bill_id
                                    FROM Roll_calls
                                   GROUP BY B.bill_id),
                     Most_sponsors AS (SELECT MAX(bill_count) as bil
```

```
                                    FROM Sponsor_counts
                                   GROUP BY bill_id
                                   ORDER BY MAX(bill_count) D
                                   LIMIT 1),
                        Other_than_most AS (SELECT MAX(S.bill_count) as
                                            FROM Sponsor_Counts
                                           WHERE S.bill_count !
                                           GROUP BY S.bill_id
                                           ORDER BY MAX(S.bill_
                                           LIMIT 1)
```

Execute Minimize Output

Legislator SQL Queries 40 points

In this section, you will write SQL queries for the North Carolina legislature schema above. Your queries will be tested immediately against **two** different databases. The second database has small modifications to prevent hard coding answers. If your query's output matches the expected output, the displayed answers will be outlined in green. To get any credit, you need green, highlighted output from the query of **both** databases. Your actual score will be determined when your query is tested against a different database but **two** green feedbacks should mean that you are on track to receive full credit.

Finally, there is an identical note before each question which should remind you of North Carolina civics. The idea is to help you avoid setting the party, or chamber to a misspelled value.

NOTE on North Carolina civics: North Carolina House of Representatives legislators have the attribute role='Representative' in the Legislators relation while roll calls in the North Carolina House of Representatives have the attribute chamber='House' in the Roll_calls relation.

Start.Name.2: List all the names of Republican legislators in the North Carolina House who have names starting with the letter H.
List the names in alphabetical order.

Execute 5 points Minimize Output

NOTE on North Carolina civics: North Carolina House of Representatives legislators have the attribute role='Representative' in the Legislators relation while roll calls in the North Carolina House of Representatives have the attribute chamber='House' in the Roll_calls relation.

Average.Number.nay.2: What is the average number of nay votes for all the roll calls in the North Carolina House?
HINT: There are two ways to compute this. One is to count up all the nay in `Votes` relation and divide by the number of roll calls. **OR** the `Roll_calls` relation already has some aggregate fields to facilitate the computation.

Execute 5 points Minimize Output

NOTE on North Carolina civics: North Carolina House of Representatives legislators have the attribute role='Representative' in the Legislators relation while roll calls in the North Carolina House of Representatives have the attribute chamber='House' in the Roll_calls relation.

Bills.Most.Roll.Calls.2: For each bill there are vote sessions (roll calls) in either or both chambers (House and Senate). List the number of roll calls and bill title(s) for each bill in the North Carolina House.
Only list the bills that have at least seven roll calls.

Execute 15 points Minimize Output

NOTE on North Carolina civics: North Carolina House of Representatives legislators have the attribute role='Representative' in the Legislators relation while roll calls in the North Carolina House of Representatives have the attribute chamber='House' in the Roll_calls relation.

Bills.Most.Sponsors.2: Give the unique names of Republican Representative(s) who sponsored the bill(s) with the most number of roll calls in the North Carolina House.

Execute 15 points Minimize Output

Sailors Database Schema

Here are the tables you'll find for the database used in the final exam. 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 Sailors (sid INTEGER PRIMARY KEY,
     sname TEXT,
     rating INTEGER,
     age INTEGER)
CREATE TABLE Boats (bid INTEGER PRIMARY KEY,
     bname TEXT,
     color TEXT)
CREATE TABLE Reserves (sid INTEGER,
     bid INTEGER,
     day TEXT, -- yyyy-mm-dd
     FOREIGN KEY (sid) REFERENCES Sailors(sid)
     ON DELETE CASCADE
     FOREIGN KEY (bid) REFERENCES Boats(bid)
     ON DELETE CASCADE)
```

Sailor Scratch area

The following scratch space can be used to help develop and test queries against the **first** of the two sailor databases described above.

```
SELECT R.bid,  day
  FROM Reserves R, Sailors S
 WHERE S.sname = 'Brutus' AND
       R.sid = S.sid
  and bid=109
```

Execute Minimize Output

Sailor SQL Queries 35 points

In this section, you will write SQL queries for the sailors database schema above. Your queries will be tested immediately against **two** different databases. The second database has small modifications to prevent hard coding answers. If your query's output matches the expected output, the displayed answers will be outlined in green. To get any credit, you need green, highlighted output from the query of **both** databases. Your actual score will be determined when your query is tested against a different database but **two** green feedbacks should mean that you are on track to receive full credit.

Last.Reserved.Boat.Lubber.2: For each boat reserved by Lubber, list the boat id and the last day it was reserved by Lubber.

```
select B.bid from Boats B where bname = 'Clipper' and not exists
  (select R.sid from Boats B2, Reserves R where bname = 'Clipper' and R.b
  except
  select R.sid from Reserves R where R.bid = B.bid)
  --select distinct R.sid from Reserves R where bid=103
  --select distinct count(sid) from Sailors
```

Execute 5 points Minimize Output

Last.Reserved.Boat.2: List all the boat bids, their names, and the last date they were reserved by sailor Lubber. If the boat was never reserved by Lubber, use null for the date.
List in order by decreasing bid.

Execute 15 points Minimize Output

All.Sailors.2: List all the bids of the boats named Sunfish that were reserved by all the sailors

Execute 15 points Minimize Output

Block Nested Loop Join Evaluations 15 points

In this section, you will show your knowledge of block nested loop join evaluation on the relations from the last homework. For this evaluation you will use block nested loop joins with buffers.

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.
The $\log_e$ function is available as `log`. Computing $log_x(y)$ is equivalent to `log(y) / log(x)`.
The ceiling function `ceil` is also available.
NOTE: Your input will be processed by the JavaScript eval function and returned with toPrecision of 3 digits such that 2500/13 will return 192 while 10/3 will return 3.33. Both with 3 digits precision.
Leaving the box empty or filled in will **NOT** affect your grade.

Enter an expression undefined

Let relation Authors take 1250 pages, relation Books take 1750 pages, and 630 buffers that can be used during the operation.

Join.BNL.Questions.2: Evaluate the number of **READ** I/Os for a join of $Authors \bowtie_{lastname \leq title} Books$ using a block nested loop join with 630 buffers.

Questions	Answers
Yes or no, will some of the buffers be used?	
Yes or no, will Authors be the outer loop?	
Yes or no, will Books be the outer loop?	
How many buffers are used for reading in the inner loop relation?	
How many buffers are used for the in-memory hash of the outer loop relation?	
How many buffers are used for writing out the joined tuples?	

10 points

Join.BNL.Evaluation.2: Given the most loop and buffer configuration using a block nested loop, calculate the number of **READ** IOs to complete the join.

Enter an expression 5 points

External Sort 15 points

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.
The $\log_e$ function is available as `log`. Computing $log_x(y)$ is equivalent to `log(y) / log(x)`.
The ceiling function `ceil` is also available.
NOTE: Your input will be processed by the JavaScript eval function and returned with toPrecision of 3 digits such that 2500/13 will return 192 while 10/3 will return 3.33. Both with 3 digits precision. Leaving the box empty or filled in will **NOT** affect your grade.

72*8 576

Assume that you have 1500 pages and 9 buffers. Answer the following questions

Pass.0.Runs.2: How many runs will be produced in pass zero?

Enter an expression 3 points

How.Many.Passes.2: How many passes will it take to sort the file completely?

Enter an expression 3 points

Total.IOs.2: What is the total (read and write) I/O cost of sorting the file completely?

Enter an expression 3 points

Buffers.In.Two.2: What is the minimum number of buffers required to sort the file completely in just two passes?

Enter an expression 6 points

B+ Trees 10 points

Figure 1 is a B+ Tree with index pages that can contain at most four keys and leaf pages that can contain at most four data entries.

Assume that the tree was initially empty and that the key values of 21, 22, 23, 24, 25, 26, 27, 28, and 29 are inserted in this order into the B+ tree. Use the table at the right to fill in the root's keys and the leaf data values for the B+ tree after the insertions. You must follow the DBMS book Chapter 10 algorithm for insertion into a B+ tree. (This means, do not do redistribution of nodes which was not covered in class.)

For any values that are empty, enter `null`.

Btree.After.Insertion.2: Fill in the key and data values for the resulting B+ tree after insertions.
Note: Even though, alternative 2 leaf pages include an asterisk in the notation, the pull downs do not include the asterisk.

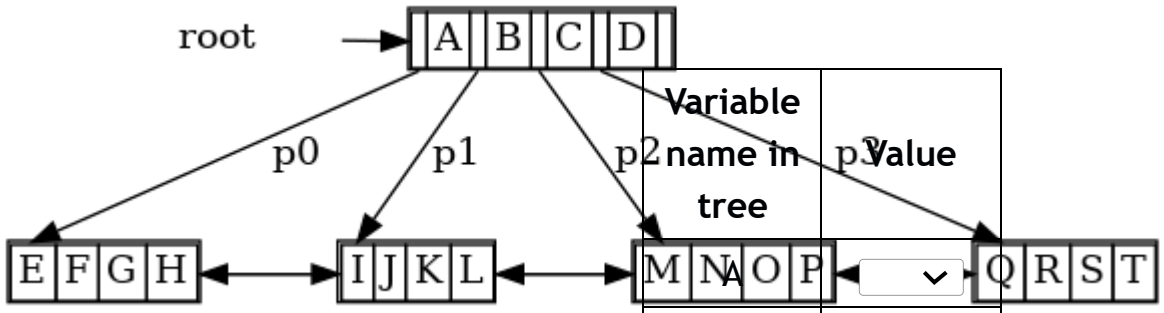


Figure 1. Initially empty with insertions into a B+ tree

Variable name in tree	Value
A	
B	
C	
D	
E	
F	
G	
H	
I	
J	
K	
L	
M	

N	
O	
P	
Q	
R	
S	
T	

10 points
