

Name: _____

Some questions will use the database instance depicted in Figure 1. The **Trees** relation indicates, for each tree, its common name, botanical name, type, typical trunk diameter (in inches), typical height (in feet), and minimum and maximum zones. The zones refer to regions of the U.S. where the trees are likely to grow well. The **Places** relation indicates, for each place, the city, state, population, zone (as above), subzone (a finer subdivision of zones), and average minimum temperature.

Trees

name	botname	ttype	dia	height	minz	maxz
White Pine	Pinus strobus	coniferous	30.0	90.0	3	8
Pitch Pine	Pinus rigida	coniferous	18.0	35.0	5	7
Bigtooth Aspen	Populus grandidentata	deciduous	15.0	70.0	3	5
Quaking Aspen	Populus tremuloides	deciduous	13.0	67.5	1	8

Places

city	state	pop	zone	subzone	mintemp
Orono	Maine	9112	5	b	-15.0
Bangor	Maine	31473	5	a	-15.1
Bar Harbor	Maine	4820	5	b	-14.0
Caribou	Maine	8312	4	a	-25.4
Van Buren	Maine	2631	3	a	-35.6
Tucson	Arizona	486699	8	a	39.0

For notational convenience in relational algebra, we shall abbreviate as follows:
Trees(name, botname, ttype, dia, height, minz, maxz) $T(N, B, T, D, H, M, X)$
Places(city, state, pop, zone, subzone, mintemp) $P(C, S, P, Z, Y, L)$

Figure 1: **Sample database instance**

1. (1 pt.) Write your name in the space provided above.

2. (39 pts.) Indicate the result of evaluating each of the following relational algebra expressions on the database instance of Figure 1.

(a) $\pi_Z P$

(b) $\pi_{NBD} \sigma_{D>15} T$

$$(c) \pi_{NBD} \sigma_{D>15} T \times \pi_S P$$

$$(d) \pi_{CSN} \sigma_{S='Arizona'} (T \underset{M \leq Z \wedge X \geq Z}{\bowtie} P)$$

3. (30 pts.) Provide relational algebra queries that return the information requested in each part below. Reminder: As in all query-writing questions, your answers should work on *all* database instances, not only the instance of Figure 1.

(a) The states of places in the database.

(b) The city and states for places in zone 5.

(c) The common names of trees that grow well in Tucson, Arizona.

4. (10 pts.) Draw query trees corresponding to each of the relational algebra expressions below. Joins should be evaluated in the order suggested by the parentheses.

$$(a) \pi_{CSS'N}((P \bowtie_{Z \geq M \wedge Z \leq X} T) \bowtie_{C=C' \wedge S \neq S' \wedge (M > Z' \vee X < Z')} \rho_{P'(C',S',P',Z',Y',L'P)})$$

$$(b) \pi_{CSS'N}((P \bowtie_{C=C' \wedge S \neq S'} \rho_{P'(C',S',P',Z',Y',L'P)}) \bowtie_{Z \geq M \wedge Z \leq X \wedge (Z' < M \vee Z' > X)} T)$$

5. (5 pts.) Suppose the query trees in your answer to Question 4 are interpreted as query plans and evaluated naively (without any optimizations). Indicate which plan is likely to be more efficient, and why. (State any assumptions you make.)

6. (15 pts.) The class notes¹ outline an analysis of the nested-loop join for an LRU buffer with space for two blocks ($C = 2$). Recall our notation from the notes: For the join $R \bowtie_{\theta} S$, we use m for the number of tuples in R and n for the number of tuples in S . The sizes of tuples of R and S , in bytes, are p and q , respectively. Blocks are B bytes in size. For the following, you may make simplifying assumptions provided they are clearly stated.
- (a) Derive the number of disk accesses made by the block nested-loop join method when the LRU buffer has space for $\lceil nq/B \rceil$ blocks. Note that a request for a disk block (such as those made on lines 3 and 5 of the pseudocode in the notes) does not result in a disk access if the requested block is in the buffer.

¹Sudarshan S. Chawathe, A Second Look at Relational Algebra, Class notes. <http://cs.umaine.edu/~chaw/> October 2006.

(b) Repeat the above analysis when the LRU buffer has space for $\lceil nq/B \rceil - 1$ blocks.

- (c) Repeat the analysis for a buffer with $\lceil nq/B \rceil - 1$ blocks, but using the following *most-recently used (MRU)* policy instead of LRU: When we need to discard a buffered block in order to make room for a newly read block, we discard the block that has been accessed most recently, breaking ties arbitrarily.