CSE 444 – Homework 4
Query Optimization

Name: ______________________________

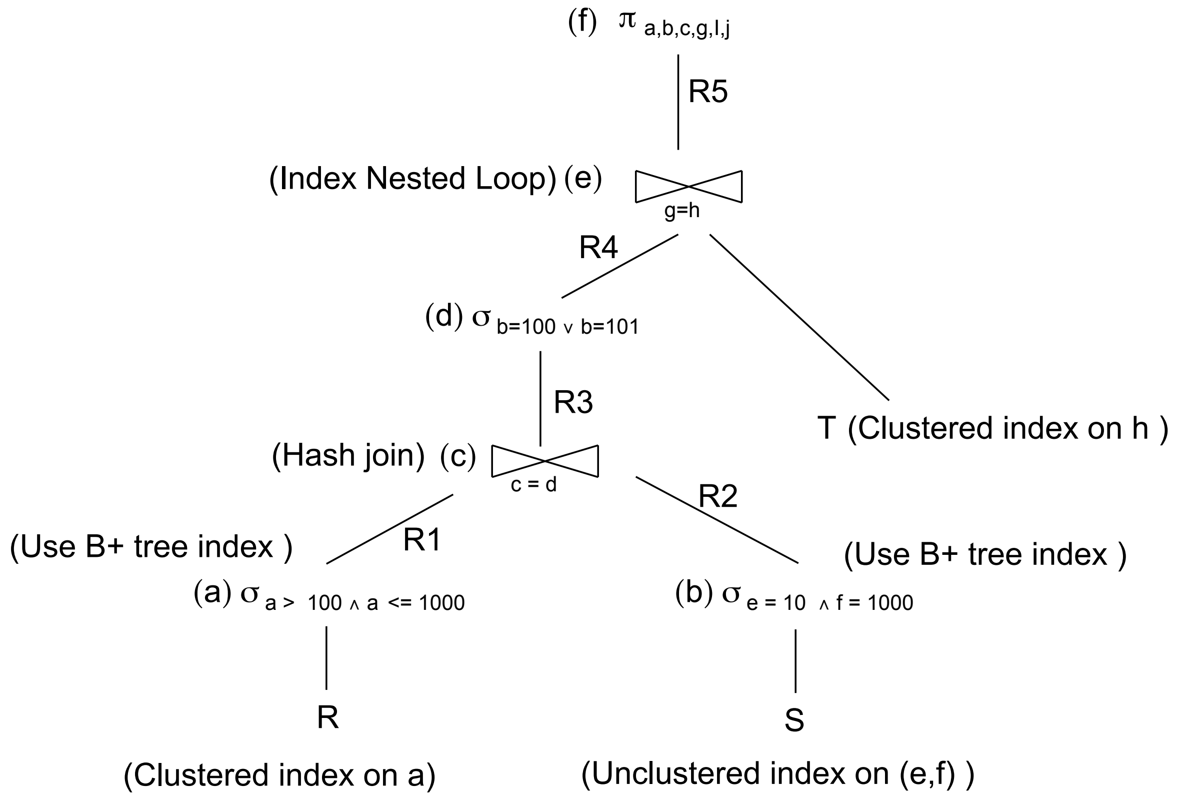
Question	Points	Score
1	20	
2	20	
Total:	40	

1 Query Plan Cost Computation

1. (20 points)

   Consider the following relations and physical query plan:

R(a,b,c)	S(d,e,f,g)	T(h,i,j)
B(R) = 1000	B(S) = 1000	B(T) = 1000
T(R) = 10,000	T(S) = 10,000	T(T) = 10,000
Min(R,a) = 0	V(S,e) = 10	V(T,h) = 100
Max(R,a) = 9000	V(S,f) = 100	
V(R,b) = 100	V(S,d) = 50	
V(R,c) = 20	V(S,g) = 40	



(a) (7 points) Compute the selectivity of the following predicates:

1. $a > 100 \wedge a \leq 1000$
2. $e = 10 \wedge f = 1000$
3. Join predicate: $c = d$
4. $b = 100 \vee b = 101$
5. $g = h$

(b) (7 points) Compute the cardinality of all intermediate relations labeled R1 through R5 and the final result, call it R6.

(c) (6 points) Compute the cost of this query plan in terms of number of pages read from disk or written to disk. Assume that all index pages are in memory at any time. Also assume that the hash table for the hash join will fit in memory.

2 Query Optimization

2. (20 points)

Consider the following SQL Query over relations R(a,b,c), S(d,e), and W(f,g,h).

```
SELECT *
FROM R, S, W
WHERE R.a = S.d
AND   R.c = W.h
```

(a) (10 points) Assume that all relations are stored in heap files, there are no indexes, only page-at-a-time nested-loop joins can be used, and the selectivity of each join predicate is 0.1%.

Show the query plan selected by a Selinger-style, bottom-up, dynamic programming optimizer. Use the number of disk IO operations as the cost function.

Hints:

- Remember that a Selinger-style optimizer will try to avoid Cartesian products in the plan. So do NOT consider such plans.
- The Selinger optimizer will only consider left-deep plans.

Draw the selected plan and show how it is derived. You can use the following table to help you but do NOT worry about computing the exact cost and size values if you don't need exact values to prune plans. In the table, P/K indicates the choice to either prune or keep the subplan. *Hint:* When joining tuples, keep in mind that the tuples get bigger.

Subquery	**Cost**	**Size of output**	**Plan**	**P/K**
R	100 page IOs	1K records on 100 pages	Sequential scan of R	K
S	1K page IOs	10K records on 1K pages	Sequential scan of S	K
W	10 page IOs	100 records on 10 pages	Sequential scan of W	K
RS	...	10K records on 2K pages	...	
SR	...		...	
...				

More space for your answer:

(b) (5 points) Consider the following small modification to the above query and consider that we add an unclustered B+ tree index on S.d as well as a clustered B+ tree index on R.b. Without re-computing the new best plan, explain how these changes affect the optimization process for this query.

```
SELECT *
FROM R, S, W
WHERE R.a = S.d
AND   R.c = W.h
AND   R.b > 100
```

(c) (5 points) What is an interesting order? Give one or two examples and explain how they can be useful in getting a better physical query plan.