BD2 – April 3rd, 2019 - Corrected
Please feel free to answer this test in English, Italian, or any mixture

1. Consider a schema R(IdR, A, B, …, IdT*), S(IdR*, IdT*,C), T(IdT, D, E,…) and the following query

   SELECT DISTINCT R.IdR, R.A, R.B, R.IdT
   FROM R, S, T
   WHERE R.IdT = S.IdT And S.IdT = T.IdT And S.C = 3

Assume that R, S and T are stored as heap files. Primary keys are R.IdR and T.IdT, while R.IdT, S.IdR and R.IdT are foreign keys. The only key for S is the set of its three attributes (S.IdR, S.IdT, S.C). Assume that unclustered RID-sorted index are defined on all the primary and foreign keys, and on S.C. Assume that the size of all indexes only depend on the number of RIDs, as indicated in the table below. Assume that every page is 4.000 bytes long, and that every attribute uses 4 bytes. Assume a buffer size of 200 pages. If you need Cardenas formula Φ(n,k), approximate it with min(n,k).

	NReg	NPag	NLeaf of Indexes	NKey	Min	Max
R	10.000.000	100.000	20.000			
S	1.000.000	5.000	2.000			
T	100.000.000	1.000.000	400.000			
Idx.S.C			See S	1000	0	100.000

a) Is DISTINCT redundant? Why?
b) Compute the selectivity factor of the three predicates in the condition
c) Compute the cost of a HashJoin plan with structure HashJoin (HashJoin(IndexFilter(S),T), R) – for simplicity does not use any projection before the end. If you need the size of a record in Join(S,T), you may compute the size of each record of S and T, as 4.000*NPag/NRec, and then do LRec(S)+LRec(T).
d) Would the use of projection before the HashJoin reduce their cost? Just explain your answer, but without performing the computation.
e) Compute the cost for an IndexNestedLoop plan with the same structure: IndexNestedLoop(IndexNestedLoop (IndexFilter(S),T), R) (draw the plan first)
f) Which of the two plans is better? Can you explain the main reason why is it cheaper in one sentence (something like: "It is better since it avoids sorting the R3 relation")?

2. Consider a relation R(X, Y, A, B, C) that contains 1.000.000 tuples, one for each pair (i,j) such that $0 < i \leq 1000$, $0 < j \leq 1000$. Assume that the relation is stored in a head file, in a completely random order, and that a combined index on (X,Y) is defined on R. The relation occupies 10.000 pages, and the combined index 4.000 pages.
   a. Consider the following condition: $100 < X \leq 110$ AND $100 < Y \leq 110$. How much does it cost to retrieve all records that satisfy it using the index? For simplicity, assume that you

first load all relevant RIDs in main memory, you sort them, and you use them to access the records.

b. How does the cost of point (a) changes if we assume that the index is clustered?
c. Assume that a B*-tree, indexed on (X,Y) like the combined index, is used to store the same relation. The B*-tree, keeps all data in the leaves and uses the intermediate nodes for a sparse index. Assuming that the B*-tree has 10.000 leaves, and each intermediate node has space for 500 children, how many intermediate nodes will have the B*-tree? How many levels will it have, leaves included?
d. How does the cost of point (a) changes if we assume that a B*-tree is used?
e. (This should be a simple question, do not do too many complex calculations) Assume that a G-tree, is used to store the same relation. Like the B*-tree, it keeps all data in the leaves and uses the intermediate nodes for a sparse index. Assuming that it divides the space in a grid of 32000 cells, how many levels would the partition tree have? And the G-tree, how many levels would it have?