This assignment is designed to make you reflect on the problems that database research focus on. It is better for you to write your own thoughts on questions 2, 3, 4, 5, and 7 than going online and checking for the best answers for your own development. You are free to do so after you come up with your own answer.

1) According to the given database schema above, write SQL queries and give the relational algebra queries of these questions.

a) (5+5 points) Show full names (Fname, Minit, Lname) and SSNs of all the employees who don't have a supervisor.

SELECT E.Fname, E.Minit, E.Lname FROM Employee E WHERE E.Super_ssn IS NULL
And the corresponding RA tree

b) (5+5 points) Show full names (Fname, Minit, Lname) and SSNs of all the employees who don't work on any projects.

SELECT E.Fname, E.Minit, E.Lname FROM Employee E WHERE E.ssn NOT IN (SELECT DISTINCT ESSN FROM WORKS_ON)

And the corresponding RA tree

c) (5+5 points) How many hours does the employee who works on the highest number of projects work?

SELECT SUM(A.hours) FROM Works_on A WHERE A.Essn IN (SELECT W.ESSN, COUNT(E.ESSN) AS projectCount FROM works_on W GROUP BY W.ESSN ORDER BY projectCount DESC LIMIT 1)

And the corresponding RA tree

d) (10 points) If you were to put two indexes on each table (except DEPT_LOCATIONS), which columns would you choose on each table? Discuss for each table in a few sentences.

To select what attributes to be indexed, you have to decide if those attributes satisfy some conditions.

- Primary Keys are most likely to be used in joins with other tables and get searched on, hence they are the strongest candidates.
- Foreign Keys are usually joined with the primary keys of the tables they belong to.
- Another good answer is that the indexing also depends on the workload: if an attribute is used very frequently in searches, that attribute is a good candidate to be indexed.

[As long as you explained your choices with one of these, you got full points.]

2) (10 points) Do you have to use indexes in the main memory? Defend your answer.

You don’t have to keep the index trees (or hash indexes) in the memory. However, to traverse the index tree, you need to bring either the whole tree to the memory or the nodes that are being traversed to the memory, so that you can make comparisons on the tree to select the required navigation pointer to the next node.
3) (10 points) Suppose you have a table that consists of 100 attributes. There are 100,000 records in this table. It is guaranteed that there is not going to be any insertions, updates or deletes to the table within the lifetime of the database. Would you prefer ISAM index or B+ tree index for this table? Defend your answer.

The correct answer is B+ tree indexes. Although it is guaranteed that there won't be any insertions/deletions/updates on the table anymore, B+ trees allow you to intrinsically perform range searches without extra cost, and gives you the sorted order of the attributes without actually needing to sort the data which are more costly in ISAM trees.

Having 100 attributes doesn't mean anything for this specific question since every index tree only works on one attribute.

[I gave 4-8 points to those who selected ISAM trees over B+ trees if you gave good enough reasons to select ISAM trees.]

4) (10 points) Why do we need different join algorithms? Explain in your own words.

Join algorithms make it possible to work under different constraints such as memory and performance constraints. While some join algorithms require more I/O and less memory, some join algorithms run faster than the others with the expense of more memory, whereas some can only be used with an index built on the join attribute.

5) (10 points) Explain what is the best case, average case and worst case for quick sort, heap sort, merge sort, insertion sort and external sort (i.e., It is the worst case for quick sort when the list is ordered). If any two or three cases are the same, explain why.

Best case, average case and worst cases are the same for heap, merge, and external sort since no matter what the distribution of the array, sorting always requires the same number of comparisons/operations. However, for insertion sort, the best case is that the numbers are ordered, in which case the algorithm run only requires one pass over the data without requiring any variable swaps. Randomly ordered arrays create the average case (still worse than the other algorithms), while reverse ordered data creates the worst case for insertion sort since the algorithm needs to swap each value to the corresponding position. For quick sort, ordered data (both ascending and descending) creates the worst case while random distributions create the average (and the best case).
6) Consider the B+ tree given in Figure 1. Using this tree, please answer the questions below individually (Not one after the other).

![B+ Tree Diagram]

**Figure 1: B+ Tree**

a) (5 points) Insert 49.

![Updated B+ Tree Diagram with Inserted 49]

b) (5 points) Insert 23.

![Updated B+ Tree Diagram with Inserted 23]

c) (5 points) Insert 90.

![Updated B+ Tree Diagram with Inserted 90]
d) (5 points) Delete 54.

7) (Bonus 10 points) Speculate on a problem (any problem) that the 6 join algorithms don’t address. Some examples are parallelism, joining more than 2 relations at once, and encrypted databases. If you can think of anything else, you’re welcome to explain it. Discuss how you can adapt one of the 6 algorithms (or a new algorithm if you can think of any) into solving the problem of your choosing.

[There are two criteria for the grading of this question:

1) Being a research topic: If the problem you stated is being considered as a research topic (or if it can be) I will give you 5 points from this criteria. If it is original but cannot be a research topic, I will give you 3 points. If it is neither a research topic nor original, you get zero points from this criteria.

2) Solution offered: If your problem statement can be considered as a research topic and you offer at least a reasonable way, you get 5 more points. If not, you can at most get 2 more points even if the solution is correct.]