TD 3 – Introduction to Apache Spark (Solutions)

Part 1 – Concept Review

Q1.
Execution Model: Hadoop executes in a strict map → reduce sequence with disk I/O between stages. Spark executes DAGs in memory.
Speed: Spark is much faster (up to 100×) for iterative tasks.
Memory: Hadoop uses disk; Spark keeps data in RAM.
Ease: Spark APIs are simpler (Python, Scala, R, Java).

Q2.
1. Resilient Distributed Dataset (RDD).
2. Lazy evaluation; action.
3. Lineage graph.
4. Partitions; parallel.

Part 2 – Understanding RDD Transformations

Q3–Q5.
Lineage graph: [rdd] → map → [rdd2] → filter → [rdd3].
Contents: rdd2 = [2,4,6,8,10,12]; rdd3 = [8,10,12]; result = [8,10,12].
Transformations: map, filter; Action: collect.

Part 3 – Key-Value Pairs

Q6. pairs = [('apple',1),('banana',1),('apple',1),('orange',1),('banana',1),('apple',1)].
Q7. reduceByKey groups by key then applies function: sum of values per key.
Q8. output = [('apple',3), ('banana',2), ('orange',1)].
Bonus: Spark reconstructs lost partitions using lineage (re-applies transformations).

Part 4 – Reflection

Q9. Lazy evaluation allows Spark to optimize execution by analyzing the full DAG before running.
Q10. Lineage graphs avoid writing intermediate results to disk and simplify recovery.
Q11. Hadoop is still useful for very large, simple batch jobs or environments with limited memory.