Tutorial 2

1. We have 12 pages of memory and we want to join tables R and S, where $[R] = 100$ and $[S] = 50$.
   (a) How many disk reads are needed to perform Chunk Nested Loops Join?
   (b) How many disk reads are needed to perform Hash-Merge Join? (Assume that no recursive partitioning is needed.)
2. Assume the relation R contains 10,000 tuples, each tuple taking up 80 bytes, and that the page size is 4 kilobytes. Further assume that we have 408 kilobytes of RAM available for storing data while executing a join algorithm. Given these assumptions, how much faster is the chunk nested loops algorithm going to be compared with the simple nested loops algorithm?
3. Alice and Bob are discussing the chunk nested loops algorithm for computing joins. Alice is claiming that allocating an output buffer the size of two pages, rather than the size of one page, is going to make the algorithm more efficient. Bob is claiming that it is going to make no difference. Is Alice right? Is Bob right? Justify your claim.
4. Alice and Bob are discussing the chunk nested loops algorithm for computing joins. Alice is claiming that the algorithm is going to run faster if we allocate an extra page-size chunk of RAM for reading the outer relation at the expense of an output buffer. Bob is claiming that it is going to slow the algorithm down. Is Alice right? Is Bob right? Justify your claim.
5. Give a scenario (tables, their schemas, their size, and a query) where an equijoin needs to be computed, and where the hash-join algorithm could not be used to process the query.

6. Suppose that tables R and S are such that both of them are sorted and, moreover, R entirely fits into RAM. Which algorithm is going to be the fastest for computing an equijoin of R and S? Justify your answer.