Milestone 4

In this milestone, we will control the shape of the join plan.

Recall that, when more than two subqueries need to be joined, this is accomplished by multiple calls to the binary join operator.

In milestone 3, we did not care about the order of these multiple calls, and therefore about the shape of the resulting join plan.

In milestone 4, it will be possible to invoke the rewriter module with two flags:

- -L stands for "left-deep" join plan, as discussed in class
  In such plans, when a join operator takes as input the result of another join operator, this may only be the left input.
  The consequence is that a join of n+1 tables leads to a plan that is a cascade of join operators.

- -B stands for "bushy" join plan.
  Here, join results may appear as both the left and the right input to a join operator. Your task is to arrange the subqueries to be joined in a bushy join plan that
    1. has smallest possible height
    2. while avoiding Cartesian products when the user query contains none of them

In real-life optimizers, bushy plans enable potentially more efficient execution because they consider plans with potentially smaller intermediate results, and because they enable parallel execution of the various plan sub-trees..

Your milestone 4 will execute the input query by

- first rewriting it to a left-deep respectively bushy query depending on the input flag,
- next executing the rewritten query. For this part, you will just reuse the execution engine from milestone 3.