Exercise

1. In this question you will consider the join of two relations from an e-commerce database. The CARTS relation represents shopping carts of items. The CONTENTS relation has the contents of all the carts, with a foreign key called cartID to CARTS:

   CARTS(cartID, customerID, date, comment).
   CONTENTS(cartID, productID, quantity, price)

   Assume that tuples in each relation are of fixed size. There are 10 CARTS tuples per page, and 100 CONTENTS tuples per page. Also assume that there are 1000 pages in the CARTS relation, and 5000 pages in the CONTENTS relation. The buffer pool is of size 1002 frames.

   You need to join on CARTS.cartID = CONTENTS.cartID.

   a) How many I/Os are required for a page-oriented nested loops join, with CARTS as the outer relation and CONTENTS as the inner relation? Assume that the buffer manager is not used (i.e. every page reference generates an I/O).

   b) Consider the same question as part (a), assuming now that the buffer manager is being used fully for this query, and that:
      - The buffer manager starts out empty
      - One output buffer frame is pinned by the join, and is used to hold output tuples until they are ready to be flushed to disk. This frame is not unpinned by the join until the end of the query.
      - One input buffer frame is pinned by the join, and holds the current page of the outer relation at all times. All I/Os to the outer relation are placed explicitly into this frame, which is not unpinned until the end of the query.
      - The buffer manager is using the MRU replacement policy
      - There is no other I/O happening in the system

      Given these assumptions, how many I/Os are required for a page-oriented nested loops join in this case?

   c) How many I/Os are required for a block nested loops join, with CONTENTS as the outer relation, CARTS as the inner relation, and 1000 pages per block?

   d) Assume now that there are only 52 frames in the buffer pool total. Rank the following join algorithms for this query in order of the number of I/Os they will perform. (You need not compute the exact number of I/Os!)
      - Hash Join, using CARTS to build the has table in the second phase
      - Sort-Merge Join
      - Block Nested Loops, CARTS as outer, block-size = 50.

2. Consider the following relational schema and SQL query:

Emp(eid,did,sal,hobby)
Dept(did,dname,floor,phone)
Finance(did,budget,sales,expences)

```
SELECT D.dname, F.budget
FROM   Emp E, Dept D, Finance F
WHERE  E.did = D.did AND D.did = F.did AND
       D.floor = 1 AND E.sal >= 59000 AND E.hobby = 'yodelling'
```

a) Identify a relational algebra expression that reflects the order of operations that a decent query optimizer would choose.
b) List the join orders that a System R query optimizer would consider. (Assume that the optimizer follows the heuristic of never considering plans that require the computation of cross-products).
c) Suppose that the following additional information is available: Unclustered B+ tree indexes exists on Emp.did, Emp.sal, Dept.did, and Finance.did (each leaf page contains up to 200 entries). The system's statistics indicate that employee salaries range from 10,000 to 60,000, employees enjoy 200 different hobbies, and the company owns two floors in the building. There are a total of 50,000 employees and 5,000 departments (each with corresponding financial information) in the database. The DBMS used by the company has just one join method available, namely index nested loops.
    i) For each of the query's base relations estimate the number of tuples that would be initially selected from that relation if all of the non-join predicates on that relation were applied to it before any join processing begins.
    ii) Given your answer to the preceding question, which of the join orders that are considered by the optimizer has the least estimated cost?