Write your name and id first. Then answer all questions.

Sorting (18 points)

(1) (18 points) Suppose that you want to sort a file containing 10,000 pages and you have 5 available buffer pages. Assume that memory is utilized well and that any available optimization to increase run size is used. Answer the following questions assuming that the sizes of input and output buffers are chosen to minimize the total number of I/Os.

1. (3 points) How many sorted runs are produced in the first pass?

2. (3 points) What is the average length of these runs?

3. (3 points) What is the number of page I/Os of the first pass?

4. (3 points) How many additional merge passes will be required to compute the fully sorted relation?
5. **(3 points)** What is the total number of page I/Os of all these additional passes?

6. **(3 points)** How many buffer pages would be required in order to sort this file in 2 passes?

### Relational Operator Implementation (36 points)

**(2) (18 points)** Suppose that a relation $R(a,b,c,d,e)$ containing 5,000,000 records, where each data page of the relation holds 10 records, is organized as a sorted file with secondary indexes. Assume that $R.a$ is a candidate key for $R$, with values lying in the range 0 to 4,999,999, and that $R$ is stored in $R.a$ order. Consider the following alternative strategies for evaluating a selection query over $R$:

- Access the sorted file for $R$ directly.
- Use a clustered B+ tree index on attribute $R.a$.
- Use a linear hashed index on attribute $R.a$.
- Use a clustered B+ tree index on attributes $(R.a,R.b)$.
- Use a linear hashed index on attributes $(R.a,R.b)$.
- Use an unclustered B+ tree index on attribute $R.b$.

For each of the following relational algebra queries, state which of the above evaluation strategies is likely to be the cheapest:

1. $\sigma_{a<50,000 \land b<50,000}(R)$

2. $\sigma_{a=50,000 \land b<50,000}(R)$
Consider the join $R \bowtie_{R.a = S.b} S$, given the following information about the relations to be joined. The cost metric is the number of page I/Os unless otherwise noted, and the cost of writing out the result should be uniformly ignored.

Relation R contains 10,000 tuples and has 10 tuples per page.
Relation S contains 2000 tuples and also has 10 tuples per page.
Attribute $b$ of relation S is the primary key for S.
Both relations are stored as simple heap files.
Neither relation has any indexes built on it.
52 buffer pages are available.

1. What is the cost of joining R and S using a page-oriented nested loops join? What is the minimum number of buffer pages required for this cost to remain unchanged?
2. What is the cost of joining R and S using a block nested loops join? What is the minimum number of buffer pages required for this cost to remain unchanged?

3. What is the cost of joining R and S using a hash join? What is the minimum number of buffer pages required for this cost to remain unchanged?

Physical Database Design (18 points)

(4) (18 points) Consider the following relations:

```
Emp(eid: integer, ename: varchar, sal: integer, age: integer, did: integer)
Dept(did: integer, budget: integer, floor: integer, mgr_eid: integer)
```

Salaries range from 100,000 to 200,000 dollars, ages vary from 50 to 55, each department has about 10 employees on average, there are 20 floors, and budgets vary from 1,000 to 10,000,000 dollars. The number of employees is 20,000, and the manager of a department may or may not be an employee in the same department. You can assume uniform distributions of values. The DBMS only supports Alternative (2) for index data entries, i.e., index data entries are always of the form (key value, data-rid).

For each of the following queries in your workload, circle the one index combination (i) through (vi)) that you would choose to create in order to speed up (only) the given query. Your choices for indexes to create is limited to those given in the question; don't consider any other alternatives.

(a) Query: Print ename, age and sal for all employees who are older than 30 and whose salary is less than 101,000. Assume that your database system will choose index-only plans whenever possible.
(i) Unclustered hash index on \( \langle \text{age}, \text{sal}, \text{ename} \rangle \) fields of Emp
(ii) Unclustered hash index on \( \langle \text{ename}, \text{age}, \text{sal} \rangle \) fields of Emp
(iii) Clustered B+ tree index on \( \langle \text{sal}, \text{age}, \text{ename} \rangle \) fields of Emp
(iv) Unclustered B+ tree index on \( \langle \text{sal}, \text{age}, \text{ename} \rangle \) fields of Emp
(v) Clustered B+ tree index on \( \langle \text{age}, \text{sal}, \text{ename} \rangle \) fields of Emp
(vi) Unclustered B+ tree index on \( \langle \text{age}, \text{sal}, \text{ename} \rangle \) fields of Emp

(b) Query: For each department with a budget over 10,000 dollars, print the name of the manager.

(i) Unclustered hash index on \text{did} field of Emp and clustered hash index on \text{budget} field of Dept
(ii) Unclustered hash index on \text{eid} field of Emp and no index on Dept
(iii) Clustered hash index on \text{did} field of Emp and unclustered B+ tree index on \text{budget} field of Dept
(iv) Clustered hash index on \text{eid} field of Emp and clustered hash index on \text{budget} field of Dept
(v) Unclustered hash index on \text{eid} field of Emp and clustered B+ tree index on \text{budget} field of Dept
(vi) Clustered hash index on \text{eid} field of Emp and unclustered B+ tree index on \text{budget} field of Dept

(c) Query: For each employee with a salary under 101,000 dollars, print the budget of the department in which he or she works.

(i) Unclustered hash index on \text{sal} for Emp and clustered B+ tree on \( \langle \text{did}, \text{budget} \rangle \) for Dept
(ii) Unclustered hash index on \text{sal} for Emp and unclustered B+ tree on \( \langle \text{did}, \text{budget} \rangle \) for Dept
(iii) Unclustered hash index on \text{sal} for Emp and unclustered B+ tree on \text{did} field of Dept
(iv) Unclustered B+ index on \text{sal} for Emp and clustered hash index on \( \langle \text{did}, \text{budget} \rangle \) for Dept
(v) Unclustered B+ index on \text{sal} for Emp and unclustered hash index on \( \langle \text{did}, \text{budget} \rangle \) for Dept
(vi) Unclustered B+ index on \text{sal} for Emp and unclustered hash index on \text{did} for Dept

Functional Dependencies and Normalization (28 points)

(5) (10 points) Suppose that we have the following three tuples in a legal instance of a relation schema \( S \) with three attributes \( ABC \) (listed in order): \( (1,2,3), (4,2,3), \) and \( (5,3,3) \).

1. Which of the following dependencies can you infer does \textit{not} hold over schema \( S \)?
   
   (a) \( A \to B \), (b) \( BC \to A \), (c) \( B \to C \)
2. Can you identify any dependencies that hold over $S$? Explain briefly.

(6) (18 points) Consider a relation $R(A,B,C,D,E)$, with FDs $A \rightarrow B$ and $C \rightarrow D$.

1. (4 points) List all keys for $R$.

2. (4 points) Is $R$ in 3NF? Explain briefly.


4. (6 points) What can you say about the decomposition of $R$ into three relations $AB$, $CD$ and $ACE$?