**Question 1**

Discuss the advantages and disadvantages of using:

1. **An unordered file:**

**Advantages:**

* Simple to implement.
* Easy and fast to insert new records as there is no need to maintain any order.
* Suitable for small datasets or where insertions are more frequent than searches.
* Disadvantages:
* Inefficient for search operations as the entire file may need to be scanned.
* Deletion can be slow and may require shifting of records.
* Not suitable for range queries as records are not stored in any particular order.

(b)  **An ordered file:**

**Advantages:**

* Efficient for search operations using binary search.
* Better performance for range queries as records are stored in a sorted manner.
* Easier to maintain order during deletion.
* Disadvantages:
* Insertion can be slow as records need to be shifted to maintain order.
* Requires additional operations to maintain the order during updates.
* Complexity increases with larger datasets.

(c) **A static hash file with fixed number of buckets and chaining employed:**

**Advantages:**

* Fast access to records using hash function.
* Efficient insertions and deletions with chaining to handle collisions.
* Good performance for equality searches.
* Disadvantages:
* Not efficient for range queries as there is no order.
* Hash function needs to be well designed to minimize collisions.
* Fixed number of buckets may lead to uneven distribution of records, causing some buckets to be overloaded.

**Operations Efficiency:**

**Unordered File:**

* **Efficient:** Insertions
* **Expensive:** Searches, Deletions, Range Queries

**Ordered File:**

* **Efficient:** Searches, Range Queries
* **Expensive:** Insertions, Updates

**Static Hash File:**

* **Efficient:** Equality Searches, Insertions, Deletions
* **Expensive:** Range Queries

**Question 2**

A data file consisting of 5632 disk blocks requires sorting using the external sort merge algorithm. The system provides a main memory buffer that can accommodate 64 blocks at a time.

1. **Determine the number of passes required in the merge phase of the external sort merge algorithm:**

Initial number of runs: 5632 / 64 = 88 runs (rounded up)

Merge phase: In each pass, the number of runs reduces by a factor of (buffer size 1) = 63.

Number of passes: log63(88) = 2 passes (since 63^2 = 3969 and 63^3 = 250047

Therefore, 2 passes are required in the merge phase.

2. **Estimate the total number of disk block accesses:**

**Initial run creation:**

Reads: 5632 blocks

Writes: 5632 blocks

Merge phase:

Pass 1: 5632 blocks read + 5632 blocks written

Pass 2: 5632 blocks read + 5632 blocks written

**Total accesses:**

Initial run creation: 5632 reads + 5632 writes = 11264 accesses

Merge phase: (5632 reads + 5632 writes) \* 2 passes = 22528 accesses

Total = 11264 + 22528 = 33792 disk block accesses.

**Question 3**

Primary, secondary, and clustering indexes:

**Primary Index:**

* Built on the primary key of the table.
* Sparse index (only one index entry per block of data).
* Used for quick lookup of rows based on the primary key.

**Secondary Index:**

* Built on non primary key attributes.
* Can be dense (one index entry per record) or sparse.
* Used for quick lookup based on non primary key attributes.

**Clustering Index:**

* Built on a non primary key attribute where the records are physically ordered in the table.
* Dense index.
* Efficient for range queries on the clustering attribute.

**Question 4**

Nested Loop Joins True/False Statements:

(a) **True.**  The choice affects I/O costs because the outer table's rows are accessed once, but the inner table's rows are accessed multiple times. Hence, using the smaller table as the outer table can reduce the number of I/O operations.

(b) **True.** Using the table that fits in memory as the inner table minimizes I/O by allowing the outer table to be read sequentially without re reading the inner table from disk.

(c) **True.** Processing on a per block basis reduces the number of I/O operations compared to per tuple processing, as multiple tuples can be processed with a single read operation.

(d) **False.** In the worst case, each block in the inner table has to be read once for each block (not each tuple) in the outer table.

(e) **False.** The choice of inner/outer table should depend on the presence of the index and not necessarily on the size of the tables.

**Question 5**

**Extend the Sort Merge Join algorithm to implement the RIGHT OUTER JOIN:**

To implement the RIGHT OUTER JOIN, you need to ensure that all records from the right table are included in the result, even if there is no matching record in the left table. The extended algorithm would look like this:

1.  **Sort Phase:**

* Sort both relations on the join attribute.

2. **Merge Phase:**

* Initialize pointers for both relations.
* While not end of either relation:
* Compare the join attributes.
* If they match, output the joined record and advance both pointers.
* If the left relation's join attribute is less than the right's, advance the left pointer.
* If the right relation's join attribute is less than the left's, output the right record with NULLs for the left relation's attributes, and advance the right pointer.
* If there are remaining records in the right relation after the end of the left relation, output each remaining right record with NULLs for the left relation's attributes.

This ensures that every record from the right relation is included in the result, fulfilling the requirement of a RIGHT OUTER JOIN.