Problem 1  (Textbook Problem 11.6)
Suppose that we are using extendable hashing on a file that contains records with the following search-key values:

2, 3, 5, 7, 11, 17, 19, 23, 29, 31

Show the extendable hash structure for this file if the hash function is $h(x) = x \mod 8$ and buckets can hold three records.

Problem 2

Given the above primary index of B+-tree of maximum fanout of 3, please answer the following questions.

1. List the nodes in the order of traversal for the following look up requests. You may label a node as a 2-tuple of the keys, e.g. $(9, \text{null})$ and $(1, 3)$.
   
   (a) Look up key 13.
   
   (b) Look up key 8.
   
   (c) Look up keys in range $[6, 16]$ (inclusive range).

2. Draw the tree after the following operations executed in the listed order. In case of leaf node split, assume two entries move to the new leaf.

   (a) Insert 6.
   
   (b) Insert 10.
   
   (c) Delete 9.

3. Assume the original tree. How many keys at most can be inserted without any internal node split at the second level of the tree (i.e. nodes $(5, \text{null})$ and $(13, 17)$)?
Problem 3

Consider the following hash buckets, each with a maximum size of 4 elements:

<table>
<thead>
<tr>
<th>catalog</th>
<th>buckets</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>8, 12, 24, 36</td>
</tr>
<tr>
<td>01</td>
<td>5, 17, 21</td>
</tr>
<tr>
<td>10</td>
<td>2, 18, 22</td>
</tr>
<tr>
<td>11</td>
<td>3, 7, 23, 31</td>
</tr>
</tbody>
</table>

Add the elements 11, 30, 28, 10 and 39 (in this order) using the following hashing strategies:

1. Use **extendible hashing** to add the elements. The buckets you start with have a global depth of 2, and each individual bucket also has a local depth of 2. If multiple entries in the directory point to the same bucket, be sure to indicate it in your solution. Please illustrate the full state of your hash buckets after each addition, including changes to the catalog and any change to global or local depth.

2. Use **linear hashing** to add the elements. Note that the "catalog" value is used purely for illustration, but you should maintain the values in your solution. The Next pointer starts by pointing at the first bucket (labeled 00), and your value for $N = 4$. Please illustrate the full state of your hash buckets after each addition, including where the Next pointer is pointing and additions to the "catalog" values.

Problem 4  (Textbook Problem 12.3)

Let relations $r_1(A,B,C)$ and $r_2(C,D,E)$ have the following properties: $r_1$ has 20,000 tuples, $r_2$ has 45,000 tuples, 25 tuples of $r_1$ fit on one block, and 30 tuples of $r_2$ fit on one block. Estimate the number of block transfers required, using each of the following join strategies for $r_1 \bowtie r_2$:

1. Nested-loop join
2. Block nested-loop join
3. Merge join
4. Hash join

Problem 5  (Textbook Problem 12.6)

Consider the following bank database, where the primary keys are underlined:

- branch(branch_name, branch_city, assets)
- customer(customer_name, customer_street, customer_city)
- loan(loan_number, branch_name, amount)
- borrower(customer_name, loan_number)
- account(account_number, branch_name, balance)
- depositor(customer_name, account_number)

Suppose that a B+Tree index on branch_city is available on relation branch, and that no other index is available. List different ways to handle the following selections that involve negation:

1. $\sigma_{\neg (\text{branch}\_\text{city}<\text{"Brooklyn"})}(\text{branch})$
2. $\sigma_{\neg (\text{branch}\_\text{city}="\text{Brooklyn"})}(\text{branch})$
3. $\sigma_{\neg (\text{branch}\_\text{city}"\text{"Brooklyn"}) \lor \text{assets}<5000}(\text{branch})$
Problem 6  (Textbook Problem 13.4)

Consider the relations \( r_1(A,B,C), r_2(C,D,E), \) and \( r_3(E,F) \), with primary keys \( A, C, \) and \( E \), respectively. Assume that \( r_1 \) has 1000 tuples, \( r_2 \) has 1500 tuples, and \( r_3 \) has 750 tuples. Estimate the size of \( r_1 \bowtie r_2 \bowtie r_3 \) and give an efficient strategy for computing the join.

Problem 7

Consider the following query on the \( account(aID, name) \) and \( deposit(aID, date, amount) \) relations:

\[
\text{SELECT a.name, d.date, d.amount} \\
\text{FROM deposit AS d} \\
\text{INNER JOIN account AS a} \\
\text{ON a.aID = d.aID} \\
\text{WHERE amount >= 400}
\]

Assume that \( account \) contains 10,000 accounts, and every account has made 50 deposits on average (the \( deposit \) table contains 500,000 deposits total). Both relations are not sorted in any particular order, and there are no indices on the relations. The in-memory buffer can hold up to 12 blocks and there is 100 tuples on average in every block (for both tables). Deposits range from $100 (inclusive) to $500 (exclusive), and you may assume an even distribution.

1. Using a \textbf{Block Nested-Loop Join}, compute the number of block accesses that will be required to perform the operation.

2. Compute the block accesses again using a \textbf{Merge-Join} instead. Both the \( deposit \) and \( account \) relations remain unordered.

3. Now let’s add some indices to these tables. First let’s add a primary index on \( deposit.amount \). The \( account \) relation remains unordered. Again, using a \textbf{Block Nested-Loop Join}, compute the number of block accesses that will be required to perform the operation. Assume that the \( WHERE \) clause is evaluated before the \textit{JOIN}, with an index fan out of 100.

4. Assume that the primary index is now changed to \( deposit.aID \), rather than \( deposit.amount \). The \( account \) relation remains unordered. Using a \textbf{Indexed Nested-Loop Join}, compute the number of block accesses that will be required to perform the operation.

5. A primary index is added on \( account.aID \). Now that both relations are sorted, recompute the number of block accesses using a \textbf{Merge-Join}.

Problem 8  (Textbook Problem 10.11)

How does the remapping of bad sectors by disk controllers affect data-retrieval rates?

Problem 9  (Textbook Problem 10.15)

Explain why the allocation of records to blocks affects database-system performance significantly.

Problem 10

Imagine that you have the following table:
### BLUEGRASS_BANDS

<table>
<thead>
<tr>
<th>bandId</th>
<th>bandName</th>
<th>ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bill Monroe</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>Grascals</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>Coon Creek Girls</td>
<td>53</td>
</tr>
<tr>
<td>4</td>
<td>Earl Scruggs</td>
<td>47</td>
</tr>
<tr>
<td>5</td>
<td>Hot Rize</td>
<td>22</td>
</tr>
<tr>
<td>6</td>
<td>Old Crow Medicine Show</td>
<td>87</td>
</tr>
<tr>
<td>7</td>
<td>Nashville Bluegrass Band</td>
<td>62</td>
</tr>
<tr>
<td>8</td>
<td>Infamous Stringdusters</td>
<td>55</td>
</tr>
<tr>
<td>9</td>
<td>Jim and Jesse</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>Doc Watson</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>Kathy Mattea</td>
<td>42</td>
</tr>
<tr>
<td>12</td>
<td>Avett Brothers</td>
<td>33</td>
</tr>
<tr>
<td>13</td>
<td>Mac Wiseman</td>
<td>92</td>
</tr>
<tr>
<td>14</td>
<td>Farmers Market</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>Psychograss</td>
<td>77</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1024</td>
<td>Tangleweed</td>
<td>63</td>
</tr>
</tbody>
</table>

The table contains 1024 rows in total. Answer the following questions about this table:

1. Suppose the database administrator decides to store `bandId` as an 8-byte integer, `bandName` as a 48-byte character array, and ranking as a 8-byte integer. If an instance of `bandName` is shorter than 48 bytes, the empty space will be filled with null characters.
   - All attributes of a tuple are stored in contiguous space within the same disk block\(^1\).
   - A disk block size is 512 bytes.
   - The disk on average performs the sequential read at 1\(\text{ms}\) per disk block and the random read at 10\(\text{ms}\) per disk block.

   (a) What is the maximum number of tuples that can be stored in a disk block?
   (b) What is the minimum number of disk blocks that need to be allocated for all tuples in the table?
   (c) What is the minimum time to read all tuples (in no particular order), assuming that the minimum number of disk blocks are allocated?

---

\(^1\)The minimum storage unit on a disk is called a *disk block* or a *disk page.*