Reminders:
   b. Rough time-estimates: 4~6 hours.
   c. Please type your answers. Illegible handwriting may get no points, at the discretion of the grader.
      Only drawings may be hand-drawn, as long as they are neat and legible.
   d. There could be more than one correct answer. We shall accept them all.
   e. Whenever you are making an assumption, please state it clearly.

Q1. B+ Tree [24 points]
Assume the following B+ tree exists with \( d = 2 \):

Sketch the state of the B+ tree after each step in the following sequence of insertions and deletions, maintaining at least 50% occupancy at each step and overflow triggered split.
In the diagram above we have not shown pointers in the leaf nodes for simplicity but remember that the leaf nodes are linked lists.

Note: Use the insertion and deletion algorithms given in the textbook section 10.5 (page 349) and 10.6 (page 353) respectively. Root node can have 1 to 2d keys. During deletion redistribute the leaf pages wherever possible.

Q1.1. (4 points) Insert 30

\[
\begin{array}{c|c|c|c}
7 & 25 & 35 & \\
\hline
3 & 4 & 5 & 6 \\
\hline
8 & 10 & 17 & 19 \\
\hline
28 & 33 & \\
\hline
42 & 51 & \\
\end{array}
\]
Q1.2.  (4 points) Insert 12

NOTE: You can push up 12 instead of 17 in this case.

Q1.3.  (4 points) Insert 60

Q1.4.  (4 points) Insert 1

NOTE: You can push up 4 instead of 5 in this case.

Q1.5.  (4 points) Delete 33

Q1.6.  (4 points) Delete 5
Q2. Bulk Loading a B+Tree [20 points]
Suppose we are bulk-loading an initially empty B+-Tree. Pages have 28 bytes to store information. A key value takes 8 bytes, and a pointer to a tree node or row takes 4 bytes. Bulk load the B+ tree with data entries 1*, 2*, ..., 12* so that each leaf is at least half full using the algorithm outlined in Section 10.8.2 (Page 360) of the textbook.

Q2.1. (4 points) What is the order of the B+ tree? Also, how many keys and pointers per node can it hold? (Note: Recall that each node in a B+ tree is a page).

Q2.2. (3 points) How many levels are in the resulting tree? (e.g. The B+Tree in Figure 1 has 2 levels).
Ans: 3 levels on the resulting tree.

Q2.3. (5 points) Sketch the final B+ tree after bulk loading. No need to show each node, just enough for us to be convinced you have the right tree.

Q2.4. (3 points) Is this the densest possible (i.e. the most filled) B+-Tree tree with these keys? If not, sketch the densest possible tree.

Q2.5. (5 points) What is the minimum number of keys that must be added so that the height of
the tree increases by 1? List these keys. Note1: There may be more than one correct answer. Note2: Please use the algorithm outlined in section 10.6 (page 353) of the textbook.

Ans: Minimum number of keys required to increase the height of the tree by 1 is: 3. They keys will all be greater than 12. Example: 13, 14, 15.

Q3. Linear Hashing [15 points]
Consider following linear hashing index to answer the questions. Assume that we split whenever a new key triggers the creation of a new overflow page.

Notes: Here h0 is (x mod 4). Currently, h0 is active, and no bucket is split. h1 is the next hash to be used (note that h1 would be (x mod 8)). Use the linear hashing algorithm outlined in Section 11.3 (page 379) of the textbook and Lecture 8: Indexes and Hashing (Linear Hashing).

Q3.1. (2 points) What is the maximum number of data entries that can be inserted in the best-case scenario, before splitting a bucket?
Ans: Maximum number of data entries that can be inserted in the best-case scenario is 4.

Q3.2. (2 points) Which bucket would 64 be inserted into (use above mentioned Figure 2 as a starting point)?
Ans: Bucket A.

Q3.3. (4 points) Show the file after inserting smallest possible single data entry whose insertion causes a bucket split and which bucket will be split (use above mentioned Figure 2 as a starting point).
Q3.4. (7 points) Show the file after inserting following entries 43, 29 (use above mentioned Figure 2 as a starting point). Assume that insertions are performed consecutively.

Ans:

<table>
<thead>
<tr>
<th>h1</th>
<th>h0</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>00</td>
</tr>
<tr>
<td>001</td>
<td>01</td>
</tr>
<tr>
<td>010</td>
<td>10</td>
</tr>
<tr>
<td>011</td>
<td>11</td>
</tr>
<tr>
<td>100</td>
<td>00</td>
</tr>
</tbody>
</table>

After inserting 43

<table>
<thead>
<tr>
<th>h1</th>
<th>h0</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>00</td>
</tr>
<tr>
<td>001</td>
<td>01</td>
</tr>
<tr>
<td>010</td>
<td>10</td>
</tr>
<tr>
<td>011</td>
<td>11</td>
</tr>
<tr>
<td>100</td>
<td>00</td>
</tr>
<tr>
<td>101</td>
<td>01</td>
</tr>
</tbody>
</table>

After inserting 29

Q4. Extendible Hashing [16 points]

Q4.1. (10 points) Consider an extendible hash structure in Table 1 below where buckets can hold up to three records. Initially the structure is empty and global depth is 2. Sketch the extendible hash structure after the records given in Table 1 (in the same order shown) have been inserted. Assume
that as mentioned in the textbook, the directory doubles in size at each overflow.

<table>
<thead>
<tr>
<th>Record</th>
<th>Hash value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>00100</td>
</tr>
<tr>
<td>b</td>
<td>00010</td>
</tr>
<tr>
<td>c</td>
<td>00001</td>
</tr>
<tr>
<td>d</td>
<td>01000</td>
</tr>
<tr>
<td>e</td>
<td>01100</td>
</tr>
<tr>
<td>f</td>
<td>10000</td>
</tr>
<tr>
<td>g</td>
<td>10111</td>
</tr>
<tr>
<td>h</td>
<td>10110</td>
</tr>
<tr>
<td>i</td>
<td>10010</td>
</tr>
<tr>
<td>j</td>
<td>11111</td>
</tr>
<tr>
<td>k</td>
<td>01000</td>
</tr>
<tr>
<td>l</td>
<td>00011</td>
</tr>
<tr>
<td>m</td>
<td>10011</td>
</tr>
<tr>
<td>n</td>
<td>10100</td>
</tr>
</tbody>
</table>

Note: Use the extendible hashing algorithm outlined in section 11.2 of the textbook.

**Ans:**

```
00
01
10
11
```

Before insertion of F

Global Depth: 3

Local Depth
Q4.2 (3 points) After sketching the extendible hashing, what are the final global and local depths of all the buckets in the hash structure?

Ans: Local depths:
- Buckets(2,3) = 2
- Buckets(1,4,5,6) = 3

Q4.3 (3 points) Suppose following range query is given “select * from table where key> x and key<=y”. Do you think this hash structure will efficiently answer the query? Why or why not?

Ans: This hash structure may not efficiently answer this question because this is a range, whereas, the hash structure has discrete buckets.

Q5. External Sorting [25 points]
Suppose you have a file with 30,000 pages and 6 available buffer pages. Answer the following questions using the general external sorting algorithm outlined in section 13.3 of the textbook. Please write the formula you used in calculating the answers.

Q5.1. (12 points) How many runs will you produce in the third pass using the general external sorting algorithm? List the number of pages that are sorted in each pass.

Ans: Pass 0 produces (30,000/6) = 5000 sorted runs of 6 pages each
Pass 1 produces (5000/5) = 1000 sorted runs of 30 pages each
Pass 2 produces (1000/5) = 200 sorted runs of 150 pages each
Pass 3 produces (200/5) = 40 sorted runs of 750 pages each
Pass 4 produces (40/5) = 8 sorted runs of 3750 pages each
Pass 5 produces (8/5) = 1 sorted run of 18750 pages and 1 sorted run of 11250 pages.
Pass 6 produces the final run.

Q5.2. (5 points) How many passes will it take to sort the file completely?
Ans: It will take 7 passes to sort the file completely.
Q5.3.  (5 points) What is the total I/O cost of sorting the file?
Ans:  \( IO\)Cost \( = 2N\left(\log_2 N \right) + 1 \)
\( = 2\times30000\times(5.47) \)
\( = 328627 \)

Q5.4.  (3 points) Now assume that we have a disk with an average seek time of 12ms, average rotation delay of 6ms and a transfer time of 2ms for each page. Assuming the cost of reading/writing a page is the sum of those values (i.e. 20ms) and do not distinguish between sequential and random disk-access – any access is 20ms, what is the total running time to sort the file in Q5.6?
Ans:  Total running time \( = 377400 \times 20 = 6572545 \)