Please follow **the most efficient** strategy to insert/delete records.

1 Hash Tables

Consider linear-probe hashing. Describe with an example why not using tombstones will incur in problems. In which operation?

2 Indexes

1. Indexes speed up query processing but it is usually a bad idea to create indexes on every attribute and every combination of attributes that might potentially be useful for an arbitrary query. Explain why?
2. Is it possible in general to have two clustering indexes on the same relation for different search keys? Explain your answer.

3 B$^+$-tree

1. Construct a B$^+$-tree for the following set of key values:

   $$(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)$$

   Assume that the tree is initially empty and values are added one-by-one in ascending order. Construct B$^+$-trees for the cases where the number of pointers that will fit in one node is as follows:

   (a) 4
   (b) 6
   (c) 8

2. For each B$^+$-tree that you have constructed, show the form of the tree after each of the following series of operations (each executed on the result of the previous):

   (a) Insert 9
   (b) Insert 10
   (c) Insert 8
   (d) Delete 23
   (e) Delete 19

4 Useful index

Consider the following relational schema, which obviously does not describe the standard situation at Aalborg University.

We assume that tutors are responsible for one or multiple study groups, students individually (not per group) hand in solutions for exercise sheets and receive individual grades in terms of the number of achieved points per sheet. Some of the tutors are more experienced (senior) than others.

student: {[sid: int, firstname: string, lastname: string, semester: int, birthdate: date]}

tutor: {[tid: int, firstname: string, lastname: string, issenior: boolean]}

studygroup: {[gid: int, tid → tutor, weekday: string, room: string, starttime: time]}

exercisesheet: {[eid: int, maxpoints: int]}

handsin: {[sid → student, eid → exercisesheet, achievedpoints: int]}

member: {[sid → student, gid → studygroup]}

For the following queries, discuss which index(es) would be useful?

Please describe all the characteristics of the index (sparse or dense? primary or secondary? hashing? multi-level?) as well as the attribute(s) used as search key in the index(es). Include any assumptions that are relevant to justify your choice.

1.

```
SELECT s.sid, s.lastname
FROM student s
    JOIN member n ON n.sid = s.sid
    JOIN studygroup g ON g.gid = n.gid
    JOIN tutor t ON t.tid = g.tid
WHERE t.issenior = false
    AND s.semester < 4;
```

2.

```
SELECT w.eid, s.sid, s.lastname
FROM student s
    JOIN handsin w ON w.sid = s.sid
    JOIN exercisesheet b ON b.eid = w.eid
WHERE w.achievedpoints = b.maxpoints;
```

3.


```sql
SELECT e.eid, a
FROM exercisesheet e LEFT OUTER JOIN (
    SELECT eid, AVG(achievedpoints) AS a
    FROM handsin
    GROUP BY eid
  ) AS averages
  ON e.eid = averages.eid;
```