Department of Computer Science

Prof. Dr. Nikolaus Augsten
Jakob-Haringer-Str. 2
5020 Salzburg, Austria
Phone: +43 662 8044 6347
E-Mail: nikolaus.augsten@plus.ac.at

PARIS LODRON UNIVERSITÄT SALZBURG

Databases II — Exam
Winter Semester 2024/25 — 16.09.2025

Name: _____ **Student ID:** _____

Hints

- Check whether you received all pages of the exam (11 pages).
- Write your name or your student ID on each sheet of the exam and hand in all pages.
- All answers are expected to be written on the exam sheets.
- Clearly highlight and enumerate additional pages that are used for longer answers. Match your text with the according exercise.
- Only use pencils that are permanent and non-red colored.
- Use the notation and techniques discussed in the lecture.
- Exercises with more than one solution are not graded.
- **Multiple-Choice**: Wrong answers result in point deductions!
- You are allowed to use one A4 sheet with your personal notes (both sides, hand written or printed).
- Exam duration: 90 minutes

Signature _____

Grading — Filled by the examiner

Exercise	1	2	3	4	5	6	7	8	9	10	Sum
Total points	1	1	1	1	1	1	1	1	1	1	10
Points reached											

Exercise 1 - *Slotted Page*. 1 Point

Consider a slotted page with the following properties:

- Size: 2^{16} bytes
- Header: densely packed
- Addressing mode: **Byte addressing** (individual bytes can be addressed)

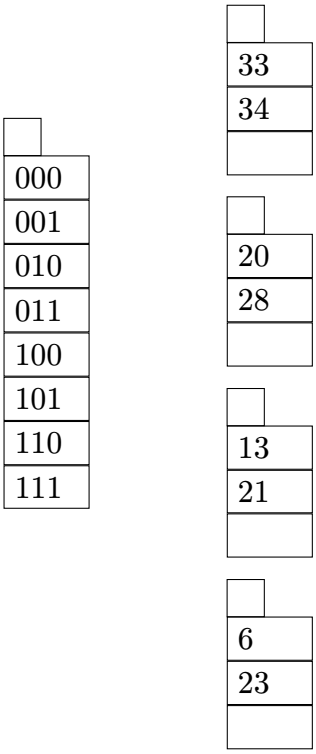
Compute the **size of each field in the header (a, f, g_i, and p_i)**. Furthermore, compute the **maximum number of tuples of size 2^5 bytes that can be stored** on a slotted page.

Exercise 2 - *Extendible Hashing*. 1 Point

The hash function $h(x)$ returns the binary values as shown in the table. **Fill in the missing values and components (global/local depth, pointer).**

Hash table:

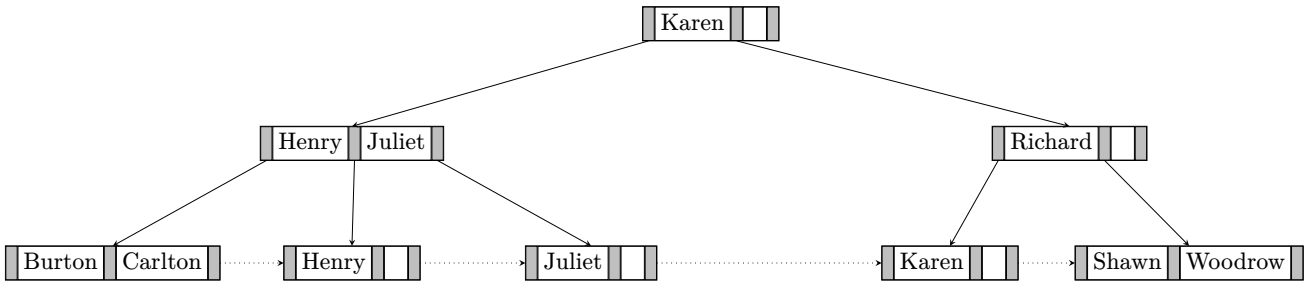
x	$h(x)$
6	1100
13	1011
20	1000
21	1010
23	1110
28	1001
33	0010
34	0110



Exercise 3 - *B⁺-Tree Deletion.* 1 Point

Consider the relation $R[id, name]$ and the following B$^+$-tree index built on attribute *name* with $m = 3$. Draw the B$^+$-tree after performing the given SQL query. Perform only as many index updates as required.

```
delete from R where name = 'Richard'
```



Exercise 4 - *Index Structures*. 1 Point

For the given table, **draw** a 3-level secondary ISAM index on attribute `Stadt`. The inner level of the index should be **dense** while both outer levels should be **sparse**. An **index block** stores **3 entries**.

`Stadt`	`KFZ`
Rom	I
London	GBM
Prag	CZ
Kiew	UA
Berlin	D
Athen	GR
Krakau	PL
Oslo	N
Dublin	IRL
Wien	A

Exercise 5 - *External Merge Sort*. 1 Point

Consider the relation $R[A]$ with $|R| = 2000$. A block can hold 2 tuples. The size of the buffer consists of 10 blocks.

How many blocks must be read/written such that an external merge sort on relation R can be performed? Do **not** count the final write step, which writes the result back to disk.

Exercise 6 - *Join Algorithms.* 1 Point

Given **two relations** R and S with the following properties:

$R[A, B, C]$:

- $|R| = 10^7$ tuples stored on $b_R = 20 \cdot 10^3$ blocks
- **dense** B$^+$-tree index on attribute A, $m = 2^8$
- **sparse** B$^+$-tree index on attribute B, $m = 2^7$
- all B$^+$-trees have **maximal height**

$S[B, D, F]$:

- $|S| = 5 \cdot 10^6$ tuples stored on $b_S = 4 \cdot 10^3$ blocks
- **single-level dense** index (ISAM) on attribute B with $5 \cdot 10^4$ blocks
- **single-level sparse** index (ISAM) on attribute D with 40 blocks

Compute the **natural join** $R \bowtie S$ based on an **index nested loop join**.

Compute the most efficient join order ($R \bowtie S$ or $S \bowtie R$) and the according **costs (number of block accesses)**. Accessing one node of the B$^+$-tree is equivalent to one block access. Duplicates do not have to be considered.

Exercise 7 - *Efficient Query Processing*. 1 Point

Consider a relation $R[A, B]$. There is a **sparse B$^+$-tree index** on attribute $R.A$ and a **dense hash index** on attribute $R.B$. Values of attribute B are unique. What is the **most efficient strategy** for processing queries of the following type?

$$\sigma_{A<a \,\vee\, B=b}(R)$$

Describe **all necessary steps**.

Exercise 8 - *Query Optimization and Join Ordering.* 1 Point

Consider the following 3 relations $R[A, B, C]$, $S[A, D, E]$, $T[A, F, G]$ and their properties:

- $|R| = 1.200$ tuples, $V(R, A) = 50$, $V(R, B) = 100$, $V(R, C) = 200$
- $|S| = 3.000$ tuples, $V(S, A) = 20$, $V(S, D) = 1.000$, $V(S, E) = 600$
- $|T| = 5.000$ tuples, $V(T, A) = 100$, $V(T, F) = 1.200$, $V(T, G) = 1.800$

Furthermore, consider the following SQL query:

```
select distinct R.A, S.D, T.G
from            R, S, T
where           R.A = S.A
                and R.A = T.A
```

a. Write the given SQL query in **algebraic normal form using an operator tree**.
b. Optimize the **operator tree** using **heuristic optimization**. The **join order** in your operator tree should be optimal (i.e., the join with the smallest intermediate result should be computed first).

Exercise 9 — 1 Point

Is the following schedule **conflict serializable**? Draw a precedence graph to verify. If it is not, explain why. If it is, give an equivalent serial schedule.

```
T1:         T2:         T3:
            write(C)
            read(A)
                        write(C)
read(B)
            write(A)
                        read(B)
            read(A)
write(B)
read(C)
                        write(B)
                        read(A)
```

Exercise 10 — 1 Point

Can the following schedule be the output of a **two-phase** locking scheduler? If so, show the schedule with all required lock and unlock instructions. Otherwise explain why. Could it be the output of a ***strict*** **two-phase** locking scheduler? Why (not)?

T1:	T2:	T3:
read(A)		
write(A)		
	read(B)	
	write(C)	
	read(A)	
	COMMIT	
		read(C)
		write(C)
		COMMIT
read(B)		
COMMIT		