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 2025/26 — 25.02.2026

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 - *Buffer Management*. 1 Point

Consider a system with a (initially empty) buffer of size 4 (i.e., the buffer can store 4 pages). **MRU** (most recently used) is used as the replacement strategy. Consider the following page access pattern of the pages A,B,C,D,E:

E A A D C E B E B A E B C A

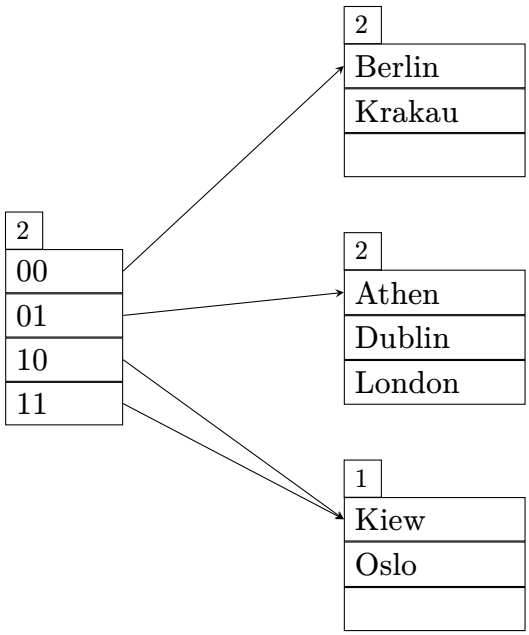
What is the number of disk accesses? Which pages are stored in the buffer after the sequence of page accesses? Complete the following table. Buffer hits are marked with "*".

buffer page 1	E												
buffer page 2		A	*										
buffer page 3				D									
buffer page 4													

Exercise 2 - *Extendible Hashing*. 1 Point

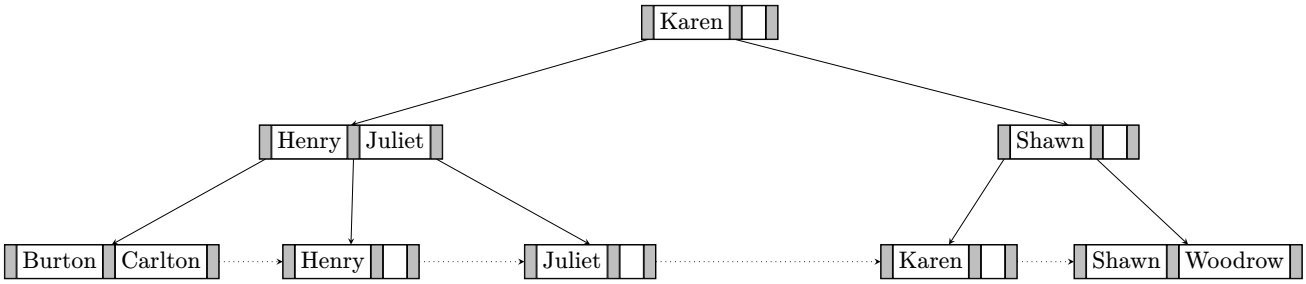
The hash function $h(x)$ returns the binary values as shown in the table. The tuple **Rom** should be inserted into the hash container. A bucket in the hash container stores up to 3 tuples. The container should be as small as possible. **Illustrate the resulting hash container**.

x	$h(x)$
Athen	0100
Berlin	0010
Dublin	0100
Kiew	1100
Krakau	0011
London	0110
Oslo	1110
Rom	0101



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

Given the following B⁺-tree with $m = 3$. Draw the B⁺-tree after deleting the key **Karen**.



Exercise 4 - *Indexes with Composite Search Keys.* 1 Point

Consider a relation $R[A, B, C, D]$ and the following four query predicates. State the order of the attributes of (up to) two ordered composite indexes that allow computing all four queries efficiently using those indexes.

(1) **WHERE** $B = 42$ **AND** $C < 100$

(2) **WHERE** $A < 50$ **AND** $B = 84$

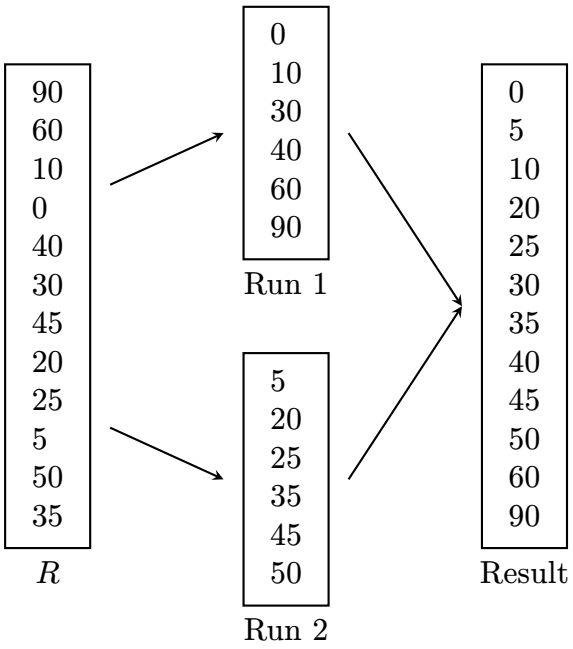
(3) **WHERE** $B = 42$ **AND** $C < 100$ **AND** $A = 100$

(4) **WHERE** $C = 100$ **AND** $D > 50$ **AND** $B = 42$

Exercise 5 - *External Merge Sort*. 1 Point

Consider the **merge phase** of **external merge sort** on the given relation $R[A]$ with 2 runs. A **block can hold 2 tuples**. The buffer can hold up to **3 blocks**.

Show all **relevant states of the buffer.** A state of the buffer is relevant if a block in the buffer is **changed completely** (i.e., all tuples are changed). For the output block, we assume that tuples in the output block are overwritten once they have been written to disk.



$\mathbf{In_1}$	$\mathbf{In_2}$	**Out**
0 10	5 20	0 5

Exercise 6 - *Join Algorithms*. 1 Point

Relation $R[\underline{A}, B, C]$ has $n_R = 3000$ tuples on $b_R = 400$ blocks and relation $S[\underline{B}, D]$ has $n_S = 2000$ tuples on $b_S = 450$ blocks.

A buffer of size $M = 30$ is used. Compute the **natural join** for the given relations based on a **hash join**.

Answer the following questions and explain:

- Which relation is used as probe/build input?
- What is the minimal number of partitions?
- Estimate the overall number of block accesses for the hash join.

Exercise 7 - *Efficient Query Processing*. 1 Point

Consider a relation $R[A]$. There is a **sparse $\mathbf{B^+}$-tree index** of attribute $R.A$ What is the **most efficient strategy** for processing **range queries** of the following type?

$$\sigma_{a<A<b}(R)$$

Describe **all necessary steps**.

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

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

- $|R| = 2000$ tuples, $V(R, A) = 100$, $V(R, C) = 20$, $V(R, D) = 80$
- $|S| = 300$ tuples, $V(S, C) = 200$, $V(S, E) = 50$
- $|T| = 500$ tuples, $V(T, B) = 20$, $V(T, C) = 50$, $V(T, F) = 30$

Furthermore, consider the following SQL query:

```
SELECT  DISTINCT R.A, S.C, T.B
FROM    R, S, T
WHERE   R.C = S.C
        AND S.C = T.C
```

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 **all** equivalent serial schedules.

T1:	T2:	T3:	T4:
		read(A)	
read(A)			
	write(C)		
read(B)			
write(A)			
			read(C)
			write(B)
	read(A)		
	write(A)		
		read(B)	

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		