Problem 1. **/ 16 P**

Instructions:

1) In this problem, you have to provide **solutions only**. You can write them right on this sheet.
2) You may use the notation, algorithms and data structures that were discussed in the lecture "Datenstrukturen & Algorithmen" without further explanations. If you use other methods, you need to **briefly** explain them in such a way that the results can be understood and checked.
3) We assume letters to be ordered alphabetically and numbers to be ordered ascendingly, according to their values.

/ 1 P a) Perform two iterations of *Insertion Sort* on the following array. The array has already been sorted by previous iterations up to the double bar.

3	7	10	15	8	6	9	5	2	13
1	2	3	4	5	6	7	8	9	10

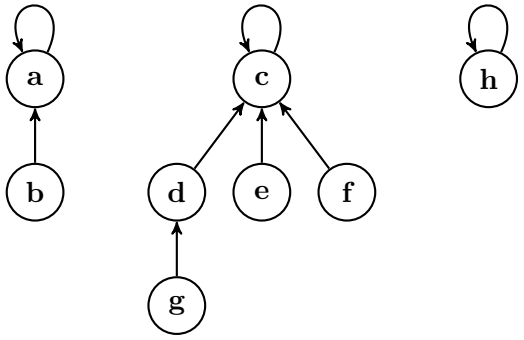
1	2	3	4	5	6	7	8	9	10

1	2	3	4	5	6	7	8	9	10

/ 1 P b) Insert the keys 20, 24, 31 and 7 in this order into the hash table below. Use open hashing with the hash function $h(k) = k \bmod 11$. Resolve collisions using quadratic probing. In case of a collision, first try probing *to the left and only after that, to the right.*

11				15	5			19	9	
0	1	2	3	4	5	6	7	8	9	10

/ 1 P c) On the following union-find data structure, execute UNION(h, FIND(g)). Use "union by height", and draw the data structure that results after the operation.



After the operation:

/ 1 P d) The following array contains the elements of a min-heap stored in the usual fashion. Specify the array after the minimum has been removed and the heap condition has been reestablished.

1	3	6	4	9	7	10	13	8
1	2	3	4	5	6	7	8	9

1	2	3	4	5	6	7	8

/ 1 P e) Specify a sequence of queries of smallest possible size that turns the list

$$A \to B \to C \to D \to E \to F$$

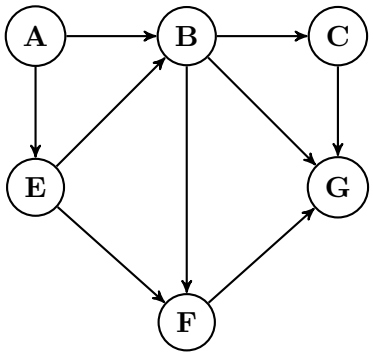
into the following list by applying the Move-to-Front rule:

$$D \to A \to C \to B \to E \to F$$

Queries: _____

/ 1 P

f) The following graph is traversed starting at vertex A. The neighbors of a vertex are visited in alphabetical order. Provide the two sequences in which a depth-first search and a breadth-first search visit the vertices.



Depth-first search:

A, ——, ——, ——, ——, ——.

Breadth-first search:

A, ——, ——, ——, ——, ——.

/ 1 P

g) Draw the natural search tree T that results if you insert the keys $5, 4, 8, 7, 1, 6, 10$ in this order in an initially empty natural search tree. Provide the preorder, postorder and inorder traversal of the keys in T.

natural search tree T:

preorder traversal:

_____

inorder traversal:

_____

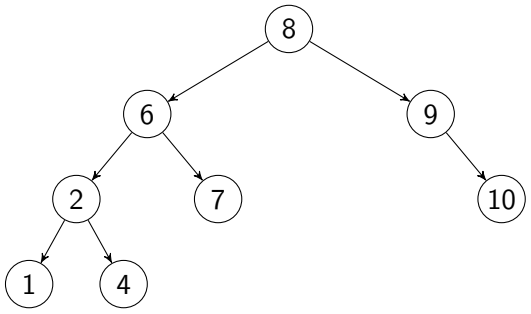
postorder traversal:

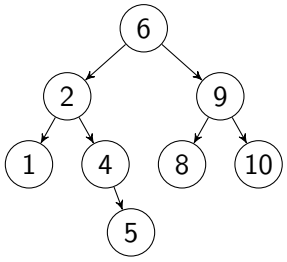
_____

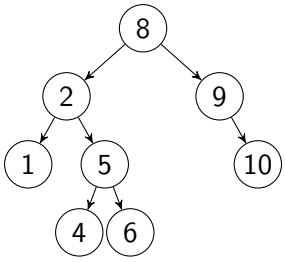
/ 2 P h) For each of the following data structures, mark with a cross whether the tree it uses is balanced (i.e. guarantees a height logarithmic in the number of stored keys) or not balanced. Every correct answer gives 0.5 points, for every wrong answer 0.5 points are removed. A missing answer gives 0 points. In overall the exercise gives at least 0 points. You do not have to justify your answers.

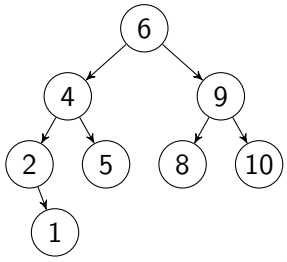
B-tree	☐ BALANCED	☐ NOT BALANCED
Heap	☐ BALANCED	☐ NOT BALANCED
AVL tree	☐ BALANCED	☐ NOT BALANCED
Natural search tree	☐ BALANCED	☐ NOT BALANCED

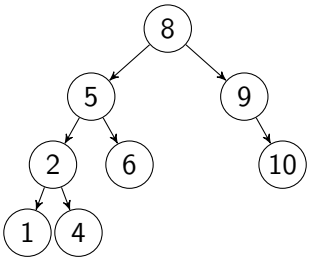
/ 1 P i) Which AVL tree results from first removing key 7 and afterwards inserting key 5 into the following AVL tree?

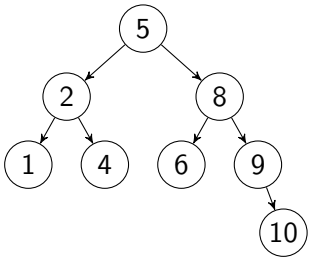


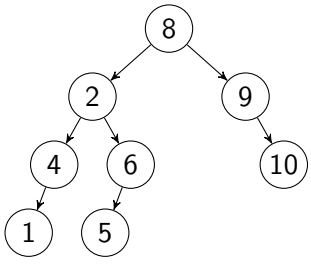
☐ 

☐ 

☐ 

☐ 

☐ 

☐ 

/ 3 P

j) Consider the following recursive formula:

$$T(n) := \begin{cases} 5 \cdot T(n/8) + 8 & n > 1 \\ 1 & n = 1 \end{cases}$$

Specify a closed (non-recursive) form for $T(n)$ that is *as simple as possible*, and prove its correctness using mathematical induction.

Hints:

(1) You may assume that n is a power of 8.
That is, use $n = 8^k$ or $k = \log_8(n)$.

(2) For $q \neq 1$, we have $\sum_{i=0}^{k} q^i = \frac{q^{k+1}-1}{q-1}$.

Derivation (if required):

Closed and simplified form:

$T(n) = T(8^k) =$

Proof by induction:

Proof by induction (continuation):

/ 1 P k) Specify (as concisely as possible) the asymptotic running time of the following code fragment in Θ notation depending on $n \in \mathbb{N}$. You do not need to justify your answer.

```
1 for ( int i = n; i >= 1; i = i - 5 ) {
2         int j = n;
3         while ( j > 1 ) {
4                 j = j / 2;
5         }
6 }
```

Running time as concisely as possible in Θ notation:

/ 1 P l) Specify (as concisely as possible) the asymptotic running time of the following code fragment in Θ notation depending on $n \in \mathbb{N}$. You do not need to justify your answer.

```
1 for ( int i = 1; i <= n; i = i + 1 ) {
2         for ( int j = 1; j < 2 * i; j = j + 2) {
3                 ;
4         }
5 }
```

Running time as concisely as possible in Θ notation:

/ 1 P m) Specify an **order** for the functions below such that the following holds: If a function f is left of a function g, then $f \in \mathcal{O}(g)$.

Example: The three functions n^3, n^7, n^9 are already in a correct order, since $n^3 \in \mathcal{O}(n^7)$ and $n^7 \in \mathcal{O}(n^9)$.

$$\frac{n^2}{\log(n)},\ \log(n^2),\ \sqrt{6^n},\ 2^{15},\ n\sqrt{n},\ \binom{n}{6},\ n\log(n)$$

Solution: _____, _____, _____, _____, _____, _____, _____.

Problem 2. **/ 10 P**

A *palindrome* is a sequence of characters whose meaning may be interpreted the same way in either forward or reverse direction, e.g. the word `RACECAR`. Formally, a palindrome is a character sequence of length 0 or 1, or a character sequence $\langle a_1, \ldots, a_l \rangle$ where $a_1 = a_l$ and $\langle a_2, \ldots, a_{l-1} \rangle$ is a palindrome.

You are given a sequence of characters $Z = \langle a_1, \ldots, a_n \rangle$ of length n. We say that Z contains a palindrome P if P is a (not necessarily contiguous) subsequence of Z. We are searching for the longest palindrome that is contained in Z.

Example: The character sequence $Z = \langle \texttt{A}, \texttt{N}, \texttt{A}, \texttt{N}, \texttt{A}, \texttt{S} \rangle$ contains the palindrome of length 0, and the palindromes `A`, `N`, `S`, `AA`, `NN`, `AAA`, `ANA`, `NAN`, `ANNA` and `ANANA` (some of these palindromes occur multiple times). The longest palindrome contained in Z therefore is `ANANA`.

For the above problem, design a *dynamic programming* algorithm. Address the following aspects in your solution.

1) What is the meaning of a table entry, and which size does the DP table have?
2) How can the table be initialized, and how can an entry be computed from the values of other entries?
3) In which order do the entries have to be computed?
4) How can the final solution be extracted once the table has been filled?

Provide also the running time of your solution, and justify your answer.

Size of the DP table / Number of entries: _____

Meaning of a table entry:

$DP[\_\_\_\_\_]$: _____

_____

Computation of an entry:

$DP[\_\_\_\_\_] =$

Order of computation:

Computing the longest contained palindrome:

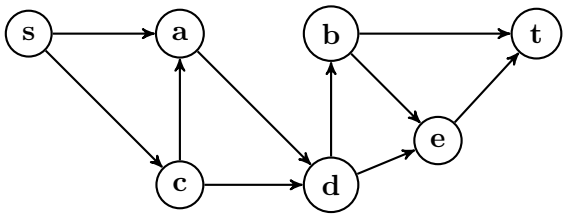
Running time (with justification):

Problem 3. **/ 7 P**

Two computers should communicate reliably over a network. The computers are not necessarily directly connected, but in general send packages over several intermediate stations through the network. We want to make sure that a computer can send packages to the other computer even if a single, arbitrary connection in the network fails.

The network is given as a set V of vertices and a set E of (directed) connections between two vertices each. Moreover, two vertices s and t in V are given that should communicate with each other.

/ 3 P a) We want to decide whether s is able to send packages to t even if a single, arbitrary connection in E is broken. For this purpose, model the above problem as a flow problem. Describe the construction of an appropriate network $N_a = (V_a, E_a, c_a)$, and describe which capacities c_a the edges should have. How can you deduce from the value of a maximum flow if the communication from s to t is always possible if a single, arbitrary connection in E is broken?



Example: In the network depicted on the left, the vertex s can send packages to vertex t even if a single, arbitrary edge is removed.

/ 4 P b) Now we want to decide whether the vertex s is able to send packages to t if a vertex in $V \setminus \{s, t\}$ is broken and cannot be used. For this purpose, model this problem as a flow problem and describe an appropriate network $N_b = (V_b, E_b, c_b)$. How can you deduce from the value of a maximum flow in this modified network if the communication from s to t is possible if only a single vertex is removed?

Example: In the network depicted above, the vertex s cannot send a package to the vertex t if vertex d cannot be used.

Subtask a)

Definition of the network N_a (if possible in words and not formally):

Vertex set V_a: _____

_____

Edge set E_a: _____

_____

Capacities c_a: _____

_____

Schematic representation / drawing of the network:

Communication from s to t is possible if: _____

_____

Subtask b)

Definition of the network N_b (if possible in words and not formally):

Vertex set V_b: _____

_____

Edge set E_b: _____

_____

Capacities c_b: _____

_____

Schematic representation / drawing of the network:

Communication from s to t is possible if: _____

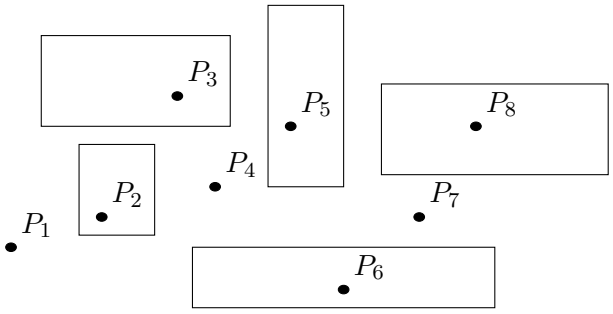
_____

/ 9 P

Problem 4.

You are given a set of n points $P_1, \ldots, P_n$ with $P_i = (x_i, y_i) \in \mathbb{Q}^2$ for all $i \in \{1, \ldots, n\}$. No two points have the same x or y coordinates. Moreover you are given a set of m axis-parallel rectangles $R_1, \ldots, R_m$ in the plane. Every rectangle R_j with $j \in \{1, \ldots, m\}$ is specified by its left lower corner point $(l_j, u_j) \in \mathbb{Q}^2$ and its right upper corner point $(r_j, o_j) \in \mathbb{Q}^2$. No two rectangles overlap or touch each other, and no point P_i is placed exactly on the border of a rectangle. We want to find all points that are not within one of the m rectangles.

Example: In the figure on the right, five rectangles and eight points are shown. The points P_1, P_4, and P_7 are not within one of the rectangles.



Design an efficient sweepline algorithm for the above problem. Address the following aspects in your solution.

1) In which direction is the sweepline moving, and what are the stopping points?
2) Which objects are maintained by the sweepline data structure, and what is an appropriate choice for this data structure?
3) What happens if the sweepline encounters a new stopping point?
4) How can the points that are not within a rectangle be recognized?
5) Provide the running time of your algorithm depending on n and m. Justify your answer.

Movement of the sweepline (choose exactly one option):

- The sweepline is a *vertical line*; it moves
  ☐ from left to right / ☐ from right to left.
- The sweepline is a *horizontal line*; it moves
  ☐ from top to bottom / ☐ from bottom to top.
- The sweepline is a *half-line*; it rotates
  ☐ in clockwise direction / ☐ in counter clockwise direction around point _____.

☐ other movement: _____

_____.

Stopping points:

set of stopping points: _____

_____

ordered by: _____

_____

Sweepline data structure:

Operations when encountering a new stopping point:

Computation of the solution:

Running time of the algorithm: