This quiz is **closed book, closed notes** and 35 minutes long

- Read the questions carefully
- No electronic equipment allowed (cell phones, tablets, computers, ...)
- Write legibly. What can’t be read will not be graded
- Use pseudocode (or English) to describe your algorithms
- Always remember to analyze the time complexity of your solution
- If you have a question about the meaning of a question, raise your hand
Problem 1. (45 points: 5 points if correct, 2.5 if unanswered, 0 if wrong)
Mark by true or false each of the following (no need to prove).

\[ \sqrt{n} \log_2 n^3 \in O(n \log_2 n) \]
\[ 9^{\log_3 n} \in \Omega(n^2 \log n) \]
\[ \log_2 2^{n^2} \in \Theta(n \log_3 3^n) \]

Radix-Sort runs in \( O(dn) \) time for an array of \( n \) \( d \)-digit integers

The following questions are on graphs; assume that \( n = |V| \) is the number of vertices, and \( m = |E| \) is the number of edges; DFS is “depth first search”; BFS is “breadth first search”; in DFS/BFS the set of edges visited during the execution of these algorithms are called tree or discovery edges; non-tree edges are the others (also called back edges in DFS, cross edges in BFS)

A directed complete graph with \( n \) nodes has exactly \( n(n - 1)/2 \) edges

Given the spanning tree \( T \) formed by the discovery (tree) edges of a BFS traversal of a connected undirected graph \( G \) started from node \( s \), for each vertex \( v \), the path on tree \( T \) is the shortest path between \( s \) and \( v \)

An edge \( e \) whose removal disconnects the graph is called a bridge; if BFS is run on a connected undirected graph \( G \), it is possible for a bridge in \( G \) to be a cross (non-tree) edge

For a connected undirected graph \( G \), the absence of back (non-tree) edges with respect to a DFS tree implies that \( G \) is acyclic

If one runs a DFS on a connected undirected graph, the number of back (non-tree) edges is exactly \( m - n + 1 \)
Problem 2. (25 points: 5 points each)

For each of the concepts listed below write a **precise** (possibly **formal**) definition. Do **not** explain or comment about the corresponding algorithm, if any.

1. topological ordering of a directed acyclic graph $G = (V, E)$

2. binary heap

3. directed cycle in a directed graph $G = (V, E)$

4. spanning tree of an undirected graph $G = (V, E)$

5. transitive closure of a directed graph $G = (V, E)$
Problem 3. (30 points)

You are given an unsorted array \( A[1 \ldots n] \) of \( n \) distinct integers. We say that \( A[i] \) is a local maximum if \( A[i] \) is bigger than its neighbors, that is, \( A[i] > A[i - 1] \) (if \( i \neq 1 \)) and \( A[i] > A[i + 1] \) (if \( i \neq n \)). For instance in \( A = \{3, 4, 1, 2, 5, 6, 0, 8, 9\} \), 4 is a local maximum, 6 is a local maximum, and 9 is a local maximum. Give an \( O(\log n) \)-time algorithm to find any of the local maximum in \( A \). If there is more than one local maximum in \( A \), we are OK with any one of them. Explain briefly how the algorithm works, and why it runs on \( O(\log n) \) time. You can assume that \( n \) is a power of two. **Hint:** If \( A[i] \) is not a local maximum because \( A[i] < A[i + 1] \), then must there be some local maximum \( A[j] \) with \( j > i \)?