This quiz is closed book, closed notes and 30 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. (42 points: 6 points if correct, 3 if unanswered, 0 if wrong)
Mark by true or false each of the following (no need to prove).

\[
\sqrt{n} \log_2 n^3 \in O(n \log_2 n) \quad \checkmark \text{True} \quad \Box \text{False}
\]

\[
9^{\log_3 n} \in \Omega(n^2 \log n) \quad \Box \text{True} \quad \checkmark \text{False}
\]

\[
\log_2 2^{n^2} \in \Theta(n \log_3 3^n) \quad \Box \text{True} \quad \Box \text{False}
\]

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 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)

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 \( v \) \quad \checkmark \text{True} \quad \Box \text{False}

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

For a connected undirected graph \( G \), the absence of back (non-tree) edges with respect to a DFS tree implies that \( G \) is acyclic \quad \checkmark \text{True} \quad \Box \text{False}

If one runs a DFS on a connected undirected graph, the number of back (non-tree) edges is exactly \( m - n + 1 \) \quad \Box \text{True} \quad \Box \text{False}
Problem 2. (24 points: 8 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. A binary heap is a nearly complete binary tree where all nodes are either greater than or equal to (or less than or equal) to each of its children.

2. A directed cycle in a directed graph $G = (V, E)$ is a set of directed edges
   \[ \{(u_1, u_2), (u_2, u_3), \ldots, (u_{l-2}, u_{l-1}), (u_{l-1}, u_l)\} \]
   where $u_l = u_1$.

3. A spanning tree of an undirected graph $G = (V, E)$ is any acyclic subgraph of $G$, i.e.,
   $T = (V, E')$ such that $E' \subseteq E$ and $T$ acyclic.
Problem 3. (34 points)

Given a sorted (low to high) array of distinct integers \( A = \{a_1, a_2, \ldots, a_n\} \), describe an \( O(\log n) \) algorithm to determine whether there exists an index \( i \) such that \( a_i = i \). For example, in \( \{-10, -3, 3, 5, 7\} \), \( a_3 = 3 \). In \( \{2, 3, 4, 5, 6, 7\} \), there is no such \( i \). Explain briefly how the algorithm works, and why your solution takes \( O(\log n) \) time.

**Answer:** The key observation is that since \( a_i \) are sorted and distinct, \( a_i - i \) is sorted as well, so we can use binary search. In fact, the algorithm works exactly like binary search.

**Algorithm** \( \text{Search} \) (\( A : \text{array}, i : \text{integer}, j : \text{integer} \))

- if \( (i = j) \) then return \( (A[i] = i) \)
- let \( k \leftarrow \lfloor (j - i)/2 \rfloor \)
- if \( (A[k] > k) \) then return \( \text{Search}(A, i, k - 1) \)
- else return \( \text{Search}(A, k, j) \)

We call the algorithm with \( \text{Search}(A, 1, n) \). The complexity is \( O(\log n) \) because it is modified binary search. Alternatively, one can solve \( T(n) = T(n/2) + 1 \).