Name (first last) .........................................................

Student ID .........................................................

- This quiz is **closed book, closed notes** and 35 minutes long
- Read the questions carefully
- No electronic equipment allowed (cell phones, PDAs, 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

1  \[\hspace{1.5cm}/45\]

2  \[\hspace{1.5cm}/25\]

3  \[\hspace{1.5cm}/30\]

Total  \[\hspace{1.5cm}/100\]
**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).

<table>
<thead>
<tr>
<th>Expression</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>$4^{\log_2 n} \in \Omega(n^2 \log n)$</td>
<td>☐</td>
<td>☒</td>
</tr>
<tr>
<td>$\log_3 3^{n^2} \in \Theta(n \log_2 2^n)$</td>
<td>☒</td>
<td>☐</td>
</tr>
<tr>
<td>$\sqrt{n} \log_3 n^2 \in O(n \log_3 n)$</td>
<td>☒</td>
<td>☐</td>
</tr>
<tr>
<td><strong>Heap-Sort</strong> runs in $O(n)$ time</td>
<td>☐</td>
<td>☒</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Statement</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>An undirected complete graph with $n$ nodes has exactly $n(n-1)/2$ edges</td>
<td>☒</td>
<td>☐</td>
</tr>
<tr>
<td>Given the spanning tree $T$ formed by the discovery (tree) edges of a DFS 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 $v$</td>
<td>☐</td>
<td>☒</td>
</tr>
<tr>
<td>An edge $e$ whose removal disconnects a graph is called a bridge; if DFS is run on a connected undirected graph $G$, every bridge in $G$ is a discovery (tree) edge in the DFS tree</td>
<td>☒</td>
<td>☐</td>
</tr>
<tr>
<td>For a connected undirected graph $G$, the presence of a back (non-tree) edge in any DFS visit of $G$ implies that $G$ has a cycle</td>
<td>☒</td>
<td>☐</td>
</tr>
<tr>
<td>If one runs a BFS on a connected undirected graph, the number of cross (i.e., non-tree) edges is exactly $m - n + 1$</td>
<td>☐</td>
<td>☒</td>
</tr>
</tbody>
</table>
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. A transitive closure of an undirected graph \( G = (V, E) \) is a graph \( G = (V, E') \) where \((u, v) \in E'\) if there is a path from \( u \) to \( v \) in \( G \).

2. The topological ordering of a directed acyclic graph \( G = (V, E) \) is an ordering of its vertices, say \( \{v_1, v_2, \ldots\} \), such that for every directed edge \((v_i, v_j) \in E\), we have that \( i < j \).

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.

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

5. 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.
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 \( j > i \)?

**Answer:** The answer to the hint is “yes”, because if \( A[i] < A[i+1] \), then the first \( j > i \) such that \( A[j] > A[j+1] \) or \( j = n \) will be a local maximum. Likewise, if \( A[i] < A[i-1] \), then there must be a local maximum to the left of \( i \). Thus, we can do binary search over \( \{1, 2, \ldots, n\} \) to find a local maximum. Here is pseudo-code:

```
Algorithm findPeak(A)
    if n = 1 then return A[1]
    i ← \lfloor n/2 \rfloor
    if A[i] < A[i+1] then return findPeak(A[i+1..n])
    if A[i] < A[i-1] then return findPeak(A[1..i-1])
    return i
```

Each recursive call takes constant time and cuts the problem size in half, so the total time is \( O(\log n) \).