FOUNDATIONS OF COMPUTING

**Instructions:** This exam consists of 8 problems, each of equal weight. Please write answers on the exam. If you would like to attach additional pages, include examinee number on each page. To keep answers concise, it is permissible to refer to algorithms in books and cite page numbers.
1.
(a) Devise an efficient implementation of the QUEUE abstract data type (procedures ENQUEUE, DEQUEUE, and EMPTYQ) based on two stacks.

*Hint: ENQUEUE on one stack and DEQUEUE from the other.*

(b) Prove that your implementation can handle any sequence of $n$ QUEUE operations in $O(n)$ time in the worst case, using, for example, the method of potential functions. (Assume that the queue is empty at the beginning of the sequence.)
2. The recursive procedure **RecMin** below returns the minimum element of an array of \( n \) items.

```plaintext
global m;

RecMin(L, n)
(1) m := L[1]
(2) RecMin(L,2,n)
(3) return m

RecMin(L,p,r)
(1) if p < r
(2) then q := floor( (p+r)/2 )
(3) RecMin(L,p,q)
(4) RecMin(L,q+1,r)
(5) else if L[p] < m
(6) then m := L[p]
```

(a) Set up a recurrence relation describing the asymptotic order of the overall worst-case running time of **RecMin** as a function of \( n \).

(b) Put the recurrence relation of part (a) in closed form. (“Solve" it.)

(c) Analyze the expected number of executions of Statement (6). Assume that the elements of the input array are distinct, and that every possible ordering has probability \( 1/n! \).
3. Which of the five problems below have been proven to be impossible to solve in polynomial time? (That is, in time no more than $cn^b$ for some finite $b$ and $c$, all natural numbers $n$, and all inputs of length $n$.) For each such impossible problem, sketch the main ideas of a proof.

(a) Euler Tour Problem
(b) Hamiltonian Path Problem
(c) Minimum Spanning Tree Problem
(d) All-Pairs Shortest Paths Problem
(e) Graph Three-Coloring Problem
4. The strongly connected components (SCCs) of a directed graph $G = (V, E)$ can be computed in time $O(|V| + |E|)$. The SCC algorithm presented in Cormen, Rivest, and Leiserson is divided into four steps.

(a) The second step of the SCC algorithm is the computation of $G^T$ from $G$, where $G^T$ is the directed graph obtained from $G$ by reversing the direction of each edge. Show how to compute the adjacency list representation of $G^T$ from the adjacency list representation of $G$ in time $O(|V| + |E|)$.

(b) The third step of the SCC algorithm requires a depth-first search of $G^T$, but in the main loop of depth-first search, the vertices must be visited in decreasing order of finishing time. (Note that finishing times are integers in the range $1, \ldots, 2|V|$.) Show how to rewrite the main loop of depth-first search to accomplish this while keeping the total time within $O(|V| + |E|)$.

(c) A source in a digraph is a vertex from which all other vertices are reachable. Suppose that $z$ is the vertex of digraph $G$ with largest finishing time during a depth-first search. Show that if $z$ is not a source in $G$ then no other vertex of $G$ can be a source.
5.
(a) Design a deterministic finite automaton, $M$, for the language $L$ consisting of all strings $x \in \{a, b\}^*$ satisfying both

(i) $x$ begins with $ab$ and

(ii) $x$ does not contain $bb$ as a substring.

(b) For your machine $M$ from part (a), let $Q$ be the set of states of $M$, with $q_0$ the initial state, let $A$ be the set of accepting states, and let $\delta$ be the transition function. For each $q \in Q$, define $L(q)$ by

$$L(q) = \{ x \in \{a, b\}^* \mid \delta(q_0, x) = q \}.$$

For each state $q$ of $M$ describe $L(q)$ in set notation or in unambiguous English.

Note: In order for $M$ to be correct,

(i) the $L(q)$ must be disjoint,

(ii) $\bigcup_{q \in Q} L(q) = \{a, b\}^*$,

(iii) $\bigcup_{q \in A} L(q) = L$, and

(iv) for every $x \in L(q)$ and every transition $\delta(q, a) = r$, it must be the case that $xa \in L(r)$. 
6. Which of the following context-free grammars are ambiguous? If the grammar is ambiguous, prove it and come up with an unambiguous grammar for the same language. If the grammar is unambiguous, give an informal but convincing argument for your conclusion.

(a) \( S \rightarrow SS \mid a \mid b \)

(b) \( S \rightarrow aSbS \mid \lambda \)

(c) \( S \rightarrow aTb \mid aSc \mid T \)
\[ T \rightarrow aTb \mid \lambda \]

(d) \( S \rightarrow N + S \mid N * S \mid N \)
\[ N \rightarrow a \mid (S) \]