This homework must be turned in on blackboard no later than Dec. 6 (day of the exam); penalty is 10% for Dec. 6.
Please respect the following guidelines for writing pseudocode:

1. C instructions are fine. But do not write object-oriented additions. Do not declare or use any class. Declare only procedures (if necessary) and explain in words what each procedure does, and what is the use of each parameter.
2. One instruction per line
3. Match the brackets with a horizontal line
4. Number your lines
5. Write down if your array is indexed 0 . . . n − 1 or 1 . . . n.

Problem 1 Describe a method for finding both the minimum and maximum of n elements using fewer than 3n/2 comparisons between elements.
   Hint: First construct a group of candidate minimums and a group of candidate maximums.
   Note that only the number of comparisons between elements matter, not the running time.
   Present pseudocode, count the number of comparisons between elements done in the worst case, and argue your solution is correct.

Problem 2 Let T be a heap storing n keys. Assume for this problem that the heap is stored in an array A starting from 1 - that is, A[1] is the biggest key in the heap. Give an efficient algorithm for reporting all the keys in T that are greater than or equal to a given query key x (which is not necessarily in T). Note that the keys do not need to be reported (printed) in sorted order. Analyse the running time. An O(k) algorithm is needed for full grade, where k is the number of elements printed. That is, the algorithm should do a constant number of elementary operations per printed element.

Problem 3 Suppose we are given a sequence S of n elements, each of which is colored red or blue. Assuming S is represented as an array, give an in-place method for ordering S so that all the blue elements are listed before all the red elements. Can you extend your approach to three colors? In-place means that only swap operations are allowed.

Problem 4 Given an array A with n integers, a monotonically increasing subsequence is a sequence of (not necessarily consecutive) indexes \( i_1 < i_2 < \ldots < i_k \) such that \( A[i_1] < A[i_2] < \ldots < A[i_k] \); the length of subsequence is k. Given an \( O(n^2) \) algorithm to find a longest monotonically increasing subsequence. Present pseudocode, argue correctness, and analyze the running time.
   Hint: Dynamic programming.
**Problem 5** An independent set of an undirected graph $G=(V,E)$ is a subset $I$ of $V$, such that no two vertices in $I$ are adjacent. That is, if $u,v \in I$, then $(u,v) \notin E$. A **maximal independent set** $M$ is an independent set such that, if we were to add any additional vertex to $M$, then it would not be independent any longer. Every graph has a maximal independent set. (Can you see this? This question is not part of the exercise, but it is worth thinking about.) Give an efficient algorithm that computes a maximal independent set for a graph $G$. What is this method’s running time?

Present pseudocode and analyze the running time. $O(|V| + |E|)$ is achievable.

**Problem 6** Suppose we are given a weighted directed graph $G = (V,E,c)$ with costs on the edges, where every cycle has strictly positive costs, and two nodes $u,v \in V$. Give an efficient (polynomial) algorithm for computing the number of shortest $u-v$ paths in $G$. Do not attempt to list all these paths!

Present pseudocode, analyze the running time, and prove correctness.