Variations of the following problems might appear on the final exam. No solutions for this set of problems will be posted.

Problem 1 Give a pseudopolynomial algorithm for Knapsack. Strive for running time of $O(nB)$, but make sure running time is polynomial in $n$ and $B$. The Knapsack problem is defined as follows. An instance consists of $n$ items $1, 2, \ldots, n$ where item $i$ has size $s_i$ and profit $p_i$, and a knapsack size $B$ with $B \geq s_i$ for all $i = 1, 2, \ldots, n$. A feasible solution consists of a subset $Q$ of $\{1, 2, \ldots, n\}$ such that $\sum_{i \in Q} s_i \leq B$. The objective is to maximize the total profit of $Q$ - that is $\sum_{i \in Q} p_i$.

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

Problem 2 Consider a strongly connected digraph $G = (V, E)$ in which each edge $e$ has a positive integer-valued length $l(e)$ and a positive fractional cost $c(e)$. The length (resp. cost) of a path is defined as the summation of the lengths (resp. costs) of all its edges. Given a budget $B > 0$, a path is said to be $B$-restricted if its cost is at most $B$. Given a pair of vertices $s$ and $t$ and a budget $B > 0$, design a dynamic programming algorithm to compute a shortest $B$-restricted path from $s$ to $t$ in $O(|E| \text{opt})$ time, where $\text{opt}$ is the length of an optimal solution.

Problem 3 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 $G$.

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

Problem 4 Consider the following game between Andrew and Barbara: It is given an array of $n$ banknotes; and then the two players take turns: first Andrew picks a banknote from one of the ends, then Barbara chooses a banknote from one of the (remaining) ends, and then Andrew again and so on until the last banknote is picked.

The banknotes can have any positive integer value, and each player keeps the banknotes (s)he picked - their total value makes her/his payoff.

For example, if the banknote values are 9 12 1 15 3, and Andrew pick from the left, then Barbara from the left , then Andrew from the right, then Barbara from the right, then Andrew picks the last banknote, Andrew’s payoff is $9 + 3 + 1 = 13$, and Barbara’s payoff is $12 + 15 = 27$.

Now assume the banknote values are given in an array $V[1..n]$. Write a polynomial-time algorithm to chose the best (maximizing payoff) first move for Andrew, assuming Barbara also plays optimally.

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