Final

Name and ONID: _____

- The final page is left blank for use as scrap. Use the reverse sides of pages for extra working room as well. Indicate when and where this might be relevant to the grader.
- This test is closed book. You may not use anything other than a pen or pencil (pencil recommended).
- Non-alcoholic beverages are permitted.
- The test is out of 45. Questions 1, 2, and 3 are each worth 15. (There is an additional bonus question.)
- You will receive partial credit for solutions. If you get part (a) of a question wrong, you can still get credit for part (b) even if the solution to part (b) depends on part (a).

You may find the following definition useful:

SAT is a problem which determines if a given boolean formula is satisfiable (has a truth assignment to the variables such that the formula evaluates to true).

For example, the formula $\mathcal{F} = (\bar{x} \vee y \vee \bar{z}) \wedge (x \vee \bar{y} \vee z) \wedge (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y})$ is satisfied by setting y and z to true and x to false. ($\vee$ = or, $\wedge$ = and)

1. Yuckdonalds is considering opening a series of restaurants along HWY 1. From the start of HWY 1, the n possible locations are (in order) at mile post $m_1, m_2, \ldots, m_n$. The constraints are as follows:

   - At each location, Yuckdonalds may open at most one restaurant.
   - The profit from opening a restaurant at location i is p_i, where $p_i > 0$ and $i = 1, 2, \ldots, n$.
   - Any two restaurants must be at least k miles apart along the highway ($k > 0$).

   Yuckdonald's goal is to choose a set of locations, subject to these constraints, to open restaurants that will maximize profit.

   Let $T(i) = max\{p_i, max_{j<i}\{T(j) + p_i : m_i - m_j \geq k\}\}$. We propose that $T(i)$ is a recursive formula that gives the maximum profit if:

   - Yuckdonald's only considers locations $m_1, m_2, \ldots, m_i$ ($i \leq n$), and
   - chooses to open a restaurant at mile post m_i.

   (a) This recursive formula is missing a base case. What should the base case be to make the formula correct and complete?

   (b) Using induction, prove that the formula is correct. Clearly identify the different components of an inductive proof.

   (c) If you have computed $T(i)$ for every value of $i = 1, 2, \ldots, n$, what is Yuckdonald's maximum profit?

   (d) Using the principals of dynamic programming and guided by the formula $T(i)$, give pseudocode for an efficient algorithm to compute the maximum profit.

   (e) What is the asymptotic running time of your algorithm?

3. You live in a world, *Algorithmica*, in which there is a polynomial-time algorithm for the decision version of SAT. That is, there is an algorithm SAT? that runs in time $T(n) = \text{poly}(n)$ (where n is the number of variables) that, given a boolean formula $\mathcal{F}$ returns TRUE if $\mathcal{F}$ is satisfiable and FALSE otherwise.

   (a) Design an algorithm, SAT-ASS, that, given a boolean formula $\mathcal{F}$, returns a satisfying truth assignment to the variables in $\mathcal{F}$ if SAT?($\mathcal{F}$) = TRUE and FALSE if $\mathcal{F}$ if SAT?($\mathcal{F}$) = FALSE.

      (i) Give pseudocode for this algorithm using SAT? as a subroutine. (This algorithm should be poly-time in this world, *Algorithmica*.

      (ii) What is the asymptotic running time of your algorithm?

   (b) Given what you know about the problem SAT, what is true about the sets P and NP in this world, *Algorithmica*?

   (c) Draw a Venn diagram of the sets P, NP, NP-complete and NP-hard in this world, *Algorithmica*.

Bonus: Show that an otherwise polynomial-time algorithm that makes at most a constant number of calls to polynomial-time subroutines runs in polynomial time, but that a polynomial number of calls to polynomial-time subroutines may result in an exponential-time algorithm.