Write your answers in the separate answer booklet.

You have 180 minutes (after you get the answer booklet) to answer seven questions.

1. For each statement below, there are two boxes in the answer booklet labeled "Yes" and "No". Check "Yes" if the statement is ***always*** true and "No" otherwise, and give a ***brief*** (at most one short sentence) explanation of your answer. **Assume *P* ≠ *NP*.** If there is any other ambiguity or uncertainty about an answer, check "No". For example:

   - $x + y = 5$

     [X] Yes [ ] No — Suppose $x = 3$ and $y = 4$.

   - 3SAT can be solved in polynomial time.

     [ ] Yes [X] No — 3SAT is NP-hard.

   - If P = NP, then Emily can beat Super Mario Brothers in four minutes and 53 seconds.

     [X] Yes [ ] No — The hypothesis is false, so the implication is true.

   Read each statement *very* carefully; some of these are deliberately subtle!

   (a) Which of the following statements are *always* true?

   - The problem of deciding whether M accepts x for a given Turing machine M and a given string x is decidable.
   - The problem of deciding whether M accepts x within $2^{|x|}$ steps for a given Turing machine M and a given string x is decidable.
   - If $P = NP$, then all decidable problems are solvable in polynomial time.
   - If L is undecidable, then $\overline{L}$ is decidable.
   - If L is not context-free, then it is undecidable.

   (b) Suppose there is a ***polynomial-time*** many-one reduction from some language A over the alphabet $\{0, 1\}$ to some other language B over the alphabet $\{0, 1\}$. Which of the following statements are ***always*** true, assuming P ≠ NP?

   - A is a subset of B.
   - If $B \in P$, then $A \in P$.
   - If B is NP-hard, then A is NP-hard.
   - If B is regular, then A is regular.
   - If B is regular, then A is decidable.

Problems 2–7 are on the following pages.

2. ***More of the same.*** For each statement below, there are two boxes in the answer booklet labeled "Yes" and "No". Check "Yes" if the statement is ***always*** true and "No" otherwise, and give a ***brief*** (at most one short sentence) explanation of your answer. **Assume $P \neq NP$.** If there is any other ambiguity or uncertainty about an answer, check "No".

   (a) Which of the following statements are true?

   - The solution to the recurrence $T(n) = \mathbf{8}T(n/\mathbf{2}) + O(n^2)$ is $T(n) = O(n^2)$.
   - The solution to the recurrence $T(n) = \mathbf{2}T(n/\mathbf{8}) + O(n^2)$ is $T(n) = O(n^2)$.
   - Every directed acyclic graph contains at least one sink.
   - Given *any* undirected graph G, we can compute a spanning tree of G in $O(V+E)$ time using depth-first search.
   - Suppose $A[1..n]$ is an array of integers. Consider the following recursive function:

$$What(i,j) = \begin{cases} 0 & \text{if } i < 0 \text{ or } i > n \\ 0 & \text{if } j < 0 \text{ or } j > n \\ \max \left\{ \begin{array}{c} What(i, j-1) \\ What(i-1, j) \\ A[i] \cdot A[j] + What(i+1, j+1) \end{array} \right\} & \text{otherwise} \end{cases}$$

     We can memoize this function into an array $What[0..n, 0..n]$ in $O(n^2)$ time, by increasing i in the outer loop and increasing j in the inner loop.

   (b) Consider the following pair of languages:

   - DirHamPath $:= \{G \mid G \text{ is a directed graph with a Hamiltonian path}\}$
   - Acyclic $:= \{G \mid G \text{ is a directed acyclic graph}\}$

   (For concreteness, assume that in both of these languages, graphs are represented by their adjacency matrices.)

   - Acyclic $\in$ NP
   - Acyclic $\cap$ DirHamPath $\in$ P
   - DirHamPath is decidable.
   - A polynomial-time many-one reduction from DirHamPath to Acyclic would imply P=NP.
   - A polynomial-time many-one reduction from Acyclic to DirHamPath would imply P=NP.

Problems 3–7 are on the following pages.

3. We are given a set A of n points on the lower half of a circle, and a set of B of n points on the upper half of the circle. (The points are given in arbitrary order, not necessarily sorted.) We want to find a set of n pairs $S = \{(a_1, b_1), \ldots, (a_n, b_n)\}$, with $\{a_1, \ldots, a_n\} = A$ and $\{b_1, \ldots, b_n\} = B$, maximizing the total length $\ell(S) = \sum_{i=1}^{n} d(a_i, b_i)$. Here, $d(u, v)$ denotes the Euclidean distance between u and v (which you may assume can be computed in $O(1)$ time).

   The example below shows two different pairings between A (shown in black) and B (shown in white), neither of which are optimal, but the pairing on the right has a larger total length.

   (a) Consider the following greedy strategy: pick a pair $(a, b) \in A \times B$ maximizing $d(a, b)$, remove a from A and b from B, and recursively pick the remaining pairs. Show that this greedy strategy does not always correctly find an optimal solution. (Your counterexample may be given in the form of a picture.)

   (b) Let a be the leftmost point in A (i.e., with the smallest x-coordinate). Let b be the rightmost point in B (i.e., with the largest x-coordinate). Prove that any optimal pairing T^* must use the pair (a, b).
   [Hint: You may use the following geometric fact: for any 4 distinct points s, t, u, v on a circle in counterclockwise order, $d(s, u) + d(t, v) > d(t, u) + d(s, v)$.]

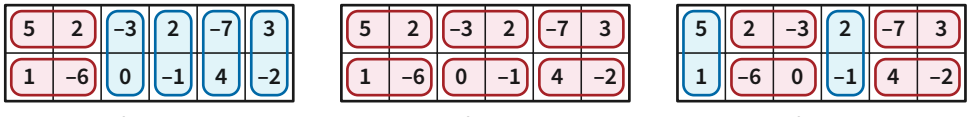
4. Suppose you are asked to tile a $2 \times n$ grid of squares with dominos (1×2 rectangles). Each domino must cover exactly two grid squares, either horizontally or vertically, and each grid square must be covered by exactly one domino.

   Each grid square is worth some number of points, which could be positive, negative, or zero. The ***value*** of a domino tiling is the sum of the points in squares covered by vertical dominos, ***minus*** the sum of the points in squares covered by horizontal dominos.

   Describe an algorithm to compute the largest possible value of a domino tiling of a given $2 \times n$ grid. Your input is an array $Points[1..2, 1..n]$ of point values.

   As an example, here are three domino tilings of the same 2×6 grid, along with their values. The third tiling is optimal; no other tiling of this grid has larger value. Thus, given this 2×6 grid as input, your algorithm should return the integer 16.

5	2	−3	2	−7	3
1	−6	0	−1	4	−2



5. ***Prove*** that the following problem (which we call MATCH) is NP-hard. The input is a finite set S of strings, all of the same length n, over the alphabet $\{0, 1, 2\}$. The problem is to determine whether there is a string $w \in \{0, 1\}^n$ such that for every string $s \in S$, the strings s and w have the same symbol in at least one position.

   For example, given the set $S = \{01220, 21110, 21120, 00211, 11101\}$, the correct output is TRUE, because the string $w = 01001$ matches the first three strings of S in the second position, and matches the last two strings of S in the last position. On the other hand, given the set $S = \{2002, 2112, 2012, 2102\}$, the correct output is FALSE.

   [Hint: Describe a reduction from SAT (or 3SAT)]

6. Describe and analyze an algorithm to determine whether the language accepted by a given DFA is finite or infinite. You can assume the input alphabet of the DFA is $\{0, 1\}$. *[Hint: DFAs are directed graphs.]*

7. Recall that a ***run*** in a string $w \in \{0, 1\}^*$ is a maximal substring of w whose characters are all equal. For example, the string 00011111110000 is the concatenation of three runs:

$$00011111110000 = 000 \bullet 1111111 \bullet 0000$$

   (a) Let L_a denote the set of all strings in $\{0, 1\}^*$ where every 0 is followed immediately by at least one 1.

   For example, L_a contains the strings 010111 and 1111 and the empty string ε, but does not contain either 001100 or 1111110.

   - Describe a DFA or NFA that accepts L_a ***and***
   - Give a regular expression that describes L_a.

   (You do not need to prove that your answers are correct.)

   (b) Let L_b denote the set of all strings in $\{0, 1\}^*$ whose run lengths are increasing; that is, every run except the last is followed immediately by a *longer* run.

   For example, L_b contains the strings 0110001111 and 1100000 and 000 and the empty string ε, but does not contain either 000111 or 100011.

   Prove that L_b is not a regular language.