UNIQNAME (print): _____

EECS 376 Midterm Exam, Winter 2024

Instructions:

This exam is closed book, closed notebook. No electronic devices are allowed. You may use one 8.5×11-inch study sheet (both sides) that you prepared yourself. The last few pages of the exam are scratch paper; you may not use your own. Make sure you are taking the exam at the time slot and location you were assigned by the staff. ***Please print your UNIQNAME at the top of each page.***

Any deviation from these rules may constitute an honor code violation. In addition, the staff reserves the right **not** to grade an exam taken in violation of this policy.

The exam consists of **8 multiple-choice** questions, **4 short-answer** questions, and **2 longer-answer** questions. For the short- and longer-answer sections, please write your answers clearly in the spaces provided. If you run out of room or need to start over, you may use the blank pages at the end, but you **MUST** make that clear in the space provided for the answer. The exam has 10 pages printed on both sides, including this page and the blank pages at the end.

Leave all pages stapled together in their original order.

Sign the honor pledge below.

Pledge:
I have neither given nor received aid on this exam, nor have I concealed any violations of the Honor Code.

I will not discuss the exam with anyone until 7pm on Thursday March 7th (once every student in the class has taken the exam, including alternate times).

I attest that I am taking the exam at the time slot and the location I was assigned by the staff.

Signature: _____

Print clearly:

Full Name: _____

Uniqname: _____

UNIQNAME (print): _____

Multiple Choice: Select all valid options.

For each of the problems in this section, select all valid options; this could be all of them, none of them, or something in between. For each option, a correct (non-)selection is worth one point. In addition, for each problem you will earn one additional point (for a total of five) if all four of your (non-)selections are correct.

1. Select **all** the **decidable** languages.
   - ◯ $L_0 = \{(\langle M \rangle, x) : \text{Turing machine } M \text{ is encoded using more than 376 bits}\}$
   - ◯ $L_1 = \{(\langle M \rangle, x) : M \text{ is a Turing machine that does not accept } x\}$
   - ◯ $L_2 = \{(\langle M \rangle, x) : M \text{ is a Turing machine that accepts } x \text{ within 12 steps}\}$
   - ◯ $L_3 = \emptyset$

2. Select **all** the **true** statements.
   - ◯ If L_1 and L_2 are undecidable languages then $L_1 \cup L_2$ is undecidable.
   - ◯ If L_1 and L_2 are undecidable languages then $L_1 \cap L_2$ is undecidable.
   - ◯ If L_1 and L_2 are decidable languages then $L_1 \cup L_2$ is decidable.
   - ◯ If L is an undecidable language then the complement of L is undecidable.

3. Select **all** the **uncountable** sets.
   - ◯ The set of rational numbers.
   - ◯ $\{\langle M \rangle : M \text{ is a Turing machine that decides some language}\}$.
   - ◯ The power set (i.e., the set of all subsets) of the integers.
   - ◯ The set of real numbers between 0 and 1.

4. Select **all** the **true** statements about shortest paths in a weighted directed graph $G = (V, E)$, where edge weights may be negative, and $s, u, v, w \in V$ denote vertices.
   - ◯ If p is a shortest path from u to v, and p' is a shortest path from v to w, then p followed by p' is a shortest path from u to w.
   - ◯ If the edges have distinct weights, then there is a unique shortest path between any two vertices in G.
   - ◯ If a shortest path from s to u *with exactly i edges* has total length d, and there is an edge $(u, v) \in E$ of weight zero, then a shortest path from s to v *with exactly $i + 1$ edges* also has total length d.
   - ◯ If a shortest path from u to w goes through v, then the u-to-v segment of that path is a shortest path from u to v.

5. Let $G = (V, E)$ be a connected, undirected graph in which all the edges have *different* weights, and let T be the *unique* minimum spanning tree of G. Let $S \subset V$ be an arbitrary subset of vertices where $S \neq V$ and $S \neq \emptyset$, and let $\partial(S) \subseteq E$ denote the subset of edges that each have one endpoint in S and the other endpoint not in S.

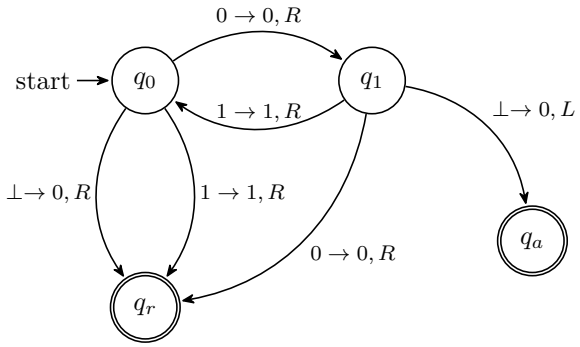
   Select **all** the **true** statements.
   - ◯ The smallest-weight edge in $\partial(S)$ *must be* in T.
   - ◯ In any cycle C of G, the smallest-weight edge *must be* in T.
   - ◯ The largest-weight edge in $\partial(S)$ *cannot be* in T.
   - ◯ In any cycle C of G, the largest-weight edge *cannot be* in T.

UNIQNAME (print): _____

Multiple Choice: Select <u>the one</u> correct option.

For each of the problems in this section, select **the one** correct option. Each one is worth 5 points; no partial credit is given.

1. Select the language (over the input alphabet $\Sigma = \{0, 1\}$) that the following TM decides, if any. Here q_a is the accept state and q_r is the reject state.



   ◯ $0^*(01)^*$
   ◯ $0^*10(0|1)^*$
   ◯ $0(10)^*$
   ◯ $00^*(10)^*$
   ◯ This Turing machine doesn't decide any language.

2. Let A, B, C be (not necessarily distinct) languages for which $(A \cap B) \leq_T C$.
   Then $A \leq_T C$ holds for ◯ all / ◯ some (but not all) / ◯ no such languages A, B, and C.

3. Consider the following algorithm:

```
1: function FUNC376(A[1, ..., n])
2:     if n = 1 then
3:         return 1
4:     else
5:         a = FUNC376(A[1, ..., ⌈2n/7⌉])
6:         b = FUNC376(A[⌈2n/7⌉, ..., ⌈4n/7⌉])
7:         c = FUNC376(A[⌈4n/7⌉, ..., ⌈6n/7⌉])
8:         d = GET203HELP(A[1, ..., n])
9:         return min(a, b, c, d)
```

   Suppose that GET203HELP has running time $O(n^{3/2})$ on an array of n elements. Select the **tightest asymptotic bound that necessarily holds** for the running time of FUNC376$(A[1, \ldots, n])$, as a function of n.

   ◯ $O(n^{3/2})$
   ◯ $O(n^{\log_{2/7} 3})$
   ◯ $O(n^{\log_{7/2} 3})$
   ◯ $O(n^{3/2} \log n)$
   ◯ $O(n^{7/2})$

UNIQNAME (print): _____

Short Answer (8 points each)

1. Let $U = \{1, \ldots, n\}$ for some given positive integer n. We are also given subsets $S_1, S_2, \ldots, S_m \subseteq U$, where each element of U belongs to at least one of the S_i. We want to select the *minimum* number of these subsets so that their union equals U, i.e., every element of U is in at least one of the selected subsets.

   Consider the following greedy algorithm: repeatedly select one of the S_i that has the *largest* number of elements that are *not in any selected subset so far* (breaking ties arbitrarily), until the union of the selected subsets is U.

   (a) **Give an explicit value of $n \leq 8$ and subsets S_i for which the greedy algorithm's output can be incorrect**, i.e., it does *not* select a minimum number of subsets. (You will give a justification in the next part.) Your solution must have $n \leq 8$ to receive any credit.

   (b) For the values you gave in the previous part, **give a valid sequence of subsets that the algorithm may select**, and **state a solution that uses fewer subsets.**

2. **Draw a DFA using five or fewer states** that decides the following language over the alphabet $\Sigma = \{3, 7, 6\}$:

   $$L = \{x \in \Sigma^* : x \text{ has an } \textbf{even} \text{ number of 3s, an } \textbf{odd} \text{ number of 7s, and } \textbf{ends} \text{ with a } 6\} .$$

   Your solution must have five or fewer states to receive any credit.

3. A string y is a *prefix* of a string $x = x_1x_2 \cdots x_\ell$ if $y = x_1x_2 \cdots x_i$ for some $0 \le i \le \ell$. For example, for the string `abcdef`, the strings `abc`, `abcdef`, and the empty string ε are some of its prefixes. (The empty string has only itself as a prefix.) Notice that a string of length ℓ has exactly $\ell + 1$ prefixes.

   Define

   $$L = \{(\langle M \rangle, x, k) : M \text{ is a Turing machine that halts on } \textit{exactly } k \text{ prefixes of } x\} .$$

   Below is some *incomplete* pseudocode of a Turing reduction H from L_{HALT} to L. (Recall that $L_{\text{HALT}} = \{(\langle M \rangle, x) : M \text{ is a Turing machine that halts on } x\}$.)

   Fill in the two missing parts of the pseudocode to make it a correct reduction; do not give any analysis. Recall that a Turing reduction may call its oracle more than once.

```
1:  Let A be an oracle ("black box") that decides L.
2:  function H(⟨M⟩, x)              ▷ Turing machine that decides L_HALT given access to A
3:      let ℓ = |x|                  ▷ ℓ ≥ 0 is the length of x
4:      for k = 0, 1, ..., ℓ + 1 do
5:          if _____ then
6:              h ← k                ▷ h is the number of prefixes of x that M halts on
7:      if h = 0 then
8:          return "reject"
9:      if x = ε then
10:         return "accept"
11:     let x' = x_1 ··· x_{ℓ−1}     ▷ x' is all but the last symbol of x
12:     return _____
```

4. Consider the following algorithm.

```
1:  A[1, ..., 376] is an array of integers in strictly increasing order: A[i] < A[j] for all 1 ≤ i < j ≤ 376.
2:  function MYSTERY(x, y)           ▷ x and y are arbitrary integers
3:      if x ≤ 0 or y ≤ 0 then
4:          return 1
5:      i ← ASKUSER(1, 100)          ▷ ask the user for an integer that must be in the range [1, 100]
6:      j ← ASKUSER(1, 100)          ▷ same as above
7:      x ← x + A[i]
8:      y ← y − A[i + j]
9:      return MYSTERY(y, x)
```

   State a potential function that can be used to prove that this algorithm terminates for *any* input integers x, y and *any* valid sequence of user inputs. **Briefly justify** (in about two sentences) why this is a valid potential function, and why it implies that the algorithm terminates.

Long Answer (14 Points Each)

1. A string made up of letters from `A`–`T` can be encoded as a string of digits from `0`–`9` using the following mapping: `A` → `0`, `B` → `1`, …, `S` → `18`, `T` → `19`. Given a string $C[1, \ldots, n]$ made up of digits from `0-9`, we wish to determine the number of ways it can be "decoded," i.e., the number of strings made up of letters from `A`–`T` that are encoded as C.

   For example, there are exactly four ways to decode the string `010194`: `ABABJE`, `ABATE`, `AKBJE`, and `AKTE`.

   For $0 \leq i \leq n$, let $d[i]$ be the number of decodings of $C[1, \ldots, i]$ (which is the empty string for $i = 0$).

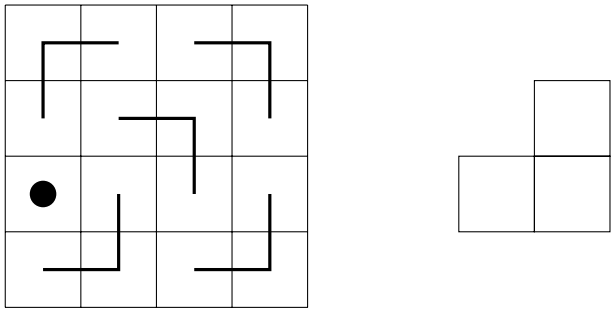
   (a) **Give, with justification, a correct recurrence and base case(s) for $d[i]$.**

   (b) **Fill in the following table for the string $C =$ `1110117101`.** Your answer will be graded for correctness based on the definition of $d[i]$, regardless of your recurrence.

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	–	1	1	1	0	1	1	7	1	0	1
$d[i]$											

2. Consider the following tiling problem: for some $n \geq 2$ that is a power of two, we want to tile an n-by-n grid of unit squares using "L-shaped" tiles that are made up of three unit squares each. However, there is *one designated* unit square in the grid that should remain *uncovered*. The rest should be covered by tiles, without any overlap and with every tile contained entirely within the grid. The tiles may be oriented arbitrarily.

   See the diagram below for a small example of a four-by-four grid and the shape of the tile. The uncovered square is marked by a circle, and the thick right angles show the orientations of the tiles in a valid tiling.



   It is not immediately obvious that there is a valid tiling for any power-of-two n and desired uncovered square; indeed, certain approaches for placing tiles may "get stuck" in unsolvable configurations. In this problem you will develop and analyze an algorithm that works in all cases.

   (a) Below is an *incomplete* algorithm for this problem, whose missing parts you will provide. The input is a square n-by-n grid where $n \geq 2$ is a power of two, and (the location of) a designated unit square to leave uncovered. The algorithm places tiles accordingly on the grid; this is its "output."

   1. *Base case (if $n = 2$):* [**MISSING PART #1**]
   2. *Recursive case (if $n > 2$):*
      (a) Designate three additional unit squares, different from the given one, to leave uncovered for now, so that:
         i. they can be covered by a single tile, and
         ii. each of the four $n/2$-by-$n/2$ *quadrants* (upper left, lower right, etc.) of the grid has exactly one square to leave uncovered.

      Specifically, choose the three squares as follows: [**MISSING PART #2**]
      (b) [**MISSING PART #3**]
      (c) Place a tile to cover the three squares that were chosen in Step 2a.

   Provide the three missing parts to make the algorithm correct. (You will justify correctness and analyze running time in the subsequent parts.)

(b) **In about 3–5 sentences, justify the correctness of the algorithm**, as you completed it in the previous part. A formal proof by induction is not required.

(c) Let $T(n)$ denote the running time of the algorithm as a function of the side length n. The cost to place a tile at a specified location, with a specified orientation, is $O(1)$.

Derive, with brief justification, an asymptotic *recurrence* and closed-form *solution* for $T(n)$; for full credit, both should be **the *tightest* possible**.

This page is intentionally left blank for scratch work.

If you have answers on this page, write "answer continues on page 9" in the corresponding solution box.

This page is intentionally left blank for scratch work.

If you have answers on this page, write "answer continues on page 10" in the corresponding solution box.