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 14 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.
   - ● $L_0 = \{(\langle M \rangle, x) :$ **Turing machine** M **is encoded using more than 376 bits**$\}$
   - ○ $L_1 = \{(\langle M \rangle, x) : M$ is a Turing machine that does not accept $x\}$
   - ● $L_2 = \{(\langle M \rangle, x) : M$ **is a Turing machine that accepts** x **within 12 steps**$\}$
   - ● $L_3 = \emptyset$

   Solution:

   - L_0 can be decided by counting the number of bits in $\langle M \rangle$ and accepting if that number is greater than 376, and rejecting otherwise.
   - L_1 is undecidable because it is the complement of the undecidable language L_{ACC}.
   - L_2 is decided by the machine that simulates running $M(x)$ for 12 steps and accepts if M accepts; otherwise, it rejects.
   - L_3 is decided by the machine that ignores its input and rejects.

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.**

   Solution:

   - For the first statement, consider L_{HALT} and $\overline{L_{\text{HALT}}}$. Their union is Σ^*, which is trivially decidable.
   - For the second statement, consider L_{HALT} and $\overline{L_{\text{HALT}}}$. Their intersection is $\emptyset$, which is trivially decidable.
   - For the third statement, we saw in class that the union of any two decidable languages is decidable.
   - For the forth statement, we saw in class that the complement of any decidable language is decidable, i.e., L is decidable if and only if $\overline{L}$ is decidable. The current statement follows by just negating both sides.

3. Select **all** the **uncountable** sets.
   - ○ The set of rational numbers.
   - ○ $\{\langle M \rangle : M$ 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.**

Solution: In class we proved that the rationals are countable.

The set of all Turing machines is countable, and therefore any subset of this set is countable as well.

In homework we proved that the power set any countably infinite set is uncountable, and in class we proved that the integers are countably infinite.

In class we proved that the set of real numbers between 0 to 1 is uncountable.

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.**

Solution:

- Consider the counter-example below. The shortest path from u to w is $u \to w$, not $u \to v \to w$.

$$\begin{tikzcd} & v \arrow[dr, "1"] & \\ u \arrow[ur, "1"] \arrow[rr, "1"] & & w \end{tikzcd}$$

- Consider the counter-example below. Both $a \to b \to d$ and $a \to c \to d$ are shortest paths from a to d.

$$\begin{tikzcd} & b \arrow[dr, "4"] & \\ a \arrow[ur, "1"] \arrow[r, "3"] & c \arrow[r, "2"] & d \end{tikzcd}$$

- Consider the counter-example below. The shortest path from s to u with exactly 1 edge has length 10, with an edge (u, v) of zero weight. But the shortest path from s to v with exactly 2 edges is $s \to w \to v$, which has length 2.

$$\begin{tikzcd} & u \arrow[dr, "0"] & \\ s \arrow[ur, "10"] \arrow[r, "1"] & w \arrow[r, "1"] & v \end{tikzcd}$$

- This fact was covered in class. Here is a proof: let P be the shortest path from u to w where $v \in P$. Let $P' \subseteq P$ be the segment of that path from u to v. Suppose for contradiction that P' is not a shortest path from u to v. Let Q' be a path from u to v shorter than P'. Then, if we replace the segment P' of P by Q', we would obtain a path from u to w that is shorter than P, which contradicts the assumption that P is the shortest path.

UNIQNAME (print): ______________________________

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.

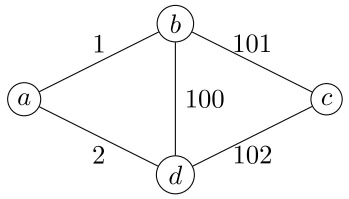
   ○ The largest-weight edge in $\partial(S)$ *cannot be* in T.

   ● **In any cycle C of G, the largest-weight edge *cannot be* in T.**

Solution: We reason about these statements using the techniques we used to prove the correctness of Kruskal's MST algorithm. Below we use *lighter* to mean "lower total weight," and *lightest* to mean "the minimum total weight". Heavier and heaviest are used in a similar way.

- Let $e = (u, v)$ be the smallest-weight edge in $\partial(S)$, and suppose for contradiction that $e \notin T$. Since T spans the graph, there is a path from u to v in T. So there must exist another edge $f \in T$ where $f \in \partial(S)$. Observe that $T \cup \{e\} \setminus \{f\}$ is a spanning tree that is lighter than T, a contradiction.

- Consider the graph below. Observe that (b, d) is the smallest-weight edge in cycle b–c–d. But (b, d) is not in the MST, which consists of the edges $(a, b), (a, d)$, and (b, c), as we can see by running Kruskal's algorithm.



- The simplest counterexample is a graph with just two vertices and an edge between them, where S consists of one of the vertices. Then that edge is the largest-weight edge in $\partial(S)$ and is also (the only edge) in T.

  A less trivial example is a path v_1–v_2–v_3 where $e = (v_1, v_2)$ has weight 1 and $e' = (v_2, v_3)$ has weight 2, and $S = \{v_2\}$. Then e' is the largest-weight edge in $\partial(S)$, and it is also in the MST.

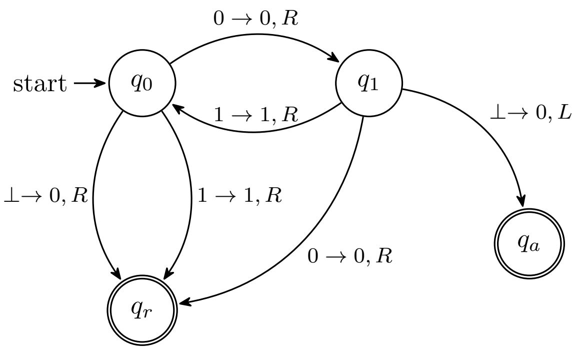
- Let $e = (u, v)$ be the largest-weight edge in a cycle C, and suppose for contradiction that $e \in T$. If we delete e from T, the two connected components of $T \setminus \{e\}$ partition the vertex set into $(S, \overline{S})$. Observe that for any edge $e' \in \partial(S)$, we have that $T \cup \{e'\} \setminus \{e\}$ is a spanning tree. Since $e \in C$ is in a cycle, there is another edge $f \in C$ where $f \in \partial(S)$. So, $T \cup \{f\} \setminus \{e\}$ is a spanning tree, and it is lighter than T because e is heavier than f—a contradiction.

UNIQNAME (print): ____

Multiple Choice: Select the one 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.

Solution: Other than one transition to the accept state, all tape movements are to the right, so the machine reads its input from left to right (but may terminate before reading all of it). To accept, the machine must be in state q_1 when it reaches end of the string. The input must start with 0 for the machine to get to q_1. After that, 10 (repeated arbitrarily many time) is the only way for the machine to get back to q_1. So, the language of the machine is $0(10)^*$.

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.

Solution: For example:

- We have that $(L_{\text{HALT}} \cap \emptyset) \leq_T \emptyset$, but $L_{\text{HALT}} \not\leq_T \emptyset$, because L_{HALT} is undecidable and $\emptyset$ is decidable.
- We have that $(\emptyset \cap \emptyset) \leq_T \emptyset$, and also that $\emptyset \leq_T \emptyset$.

3. Consider the following algorithm:

UNIQNAME (print): ____________________

```
function FUNC376($A[1, \ldots, n]$)
    if $n = 1$ then
        return 1
    else
        $a$ = FUNC376($A[1, \ldots, \lceil 2n/7 \rceil]$)
        $b$ = FUNC376($A[\lceil 2n/7 \rceil, \ldots, \lceil 4n/7 \rceil]$)
        $c$ = FUNC376($A[\lceil 4n/7 \rceil, \ldots, \lceil 6n/7 \rceil]$)
        $d$ = GET203HELP($A[1, \ldots, n]$)
        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.

- [✓ $O(n^{3/2})$]
- ○ $O(n^{\log_{2/7} 3})$
- ○ $O(n^{\log_{7/2} 3})$
- ○ $O(n^{3/2} \log n)$
- ○ $O(n^{7/2})$

Solution: The running time $T(n)$ of this algorithm satisfies the recurrence $T(n) = 3T(2n/7) + O(n^{3/2})$. This is covered by the Master Theorem, with $k = 3$, $b = 7/2$, $d = 3/2$. We have that $k < b^d = (7/2)^{3/2}$, because squaring both sides, we have that $3^2 = 9 < (7/2)^3 = 243/8$. From the Master Theorem, we get that $T(n) = O(n^d) = O(n^{3/2})$.

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.

   Solution: Let $n = 6, S_1 = \{1,2,3\}, S_2 = \{1,4\}, S_3 = \{2,5\}, S_4 = \{3,6\}$.
   Here is another, smaller example for which the algorithm *can* produce a suboptimal output, depending on how it breaks ties: $n = 4, S_1 = \{1,2\}, S_2 = \{2,3\}$, $S_3 = \{3,4\}$.

   (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.**

   Solution: The greedy algorithm selects all four subsets, starting with S_1, but an optimal solution is S_2, S_3, S_4.
   For the smaller example, The algorithm ends up taking all three subsets if it first breaks the tie to select S_2, but the optimal solution is S_1, S_3.

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.

Solution:

$$\begin{tikzcd}
 & & & \boxed{q_4} \arrow[loop right, "6"] \arrow[dll, bend right, "7"'] \arrow[ddl, bend left, "3"] \\
\text{Start} \arrow[r] & q_0 \arrow[loop above, "6"] \arrow[r, bend left, "7"] \arrow[d, bend left, "3"] & q_1 \arrow[l, bend left, "7"] \arrow[d, bend left, "3"] \arrow[ur, "6"] & \\
 & q_2 \arrow[loop left, "6"] \arrow[u, bend left, "3"] \arrow[r, bend left, "7"] & q_3 \arrow[loop right, "6"] \arrow[l, bend left, "7"] \arrow[u, bend left, "3"] &
\end{tikzcd}$$

UNIQNAME (print): ______________________________

3. A string y is a *prefix* of a string $x = x_1 x_2 \cdots x_\ell$ if $y = x_1 x_2 \cdots x_i$ 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 $H(\langle M\rangle, x)$              ▷ Turing machine that decides $L_{\text{HALT}}$ given access to $A$
3:      let $\ell = |x|$                           ▷ $\ell \ge 0$ is the length of $x$
4:      for $k = 0, 1, \ldots, \ell + 1$ do
5:          if ______ $A(\langle M\rangle, x, k)$ accepts ______ then
6:              $h \leftarrow k$                   ▷ $h$ is the number of prefixes of $x$ that $M$ halts on
7:      if $h = 0$ then
8:          return "reject"
9:      if $x = \varepsilon$ then
10:         return "accept"
11:     let $x' = x_1 \cdots x_{\ell-1}$           ▷ $x'$ is all but the last symbol of $x$
12:     return ______ $A(\langle M\rangle, x', h-1)$ ______
```

Solution:

1. For the first blank, exactly one of these calls will accept: the one where k is the exact number of prefixes of x that M halts on. So, after the for-loop is complete, h equals this number.

2. The second blank can alternatively be: "the opposite of $A(\langle M\rangle, x', h)$".

   Since M halts on exactly h prefixes of x, we have that $M(x)$ halts if and only if M halts on exactly $h - 1$ prefixes of x'. This is because x and x' have exactly the same prefixes, except that x itself is a prefix of x but not of x'.

4. Consider the following algorithm.

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

Solution: One valid potential function is $x + y$. It is valid because from one call of MYSTERY to the next, the potential decreases by at least 1: $x + y$ becomes

$$y - A[i+j] + x + A[i] = x + y - (A[i+j] - A[i]) \leq x + y - 1 \ ,$$

because $j \geq 1$ and A is an array of strictly increasing integers. Also, once $x + y \leq 0$, we have that $x \leq 0$ or $y \leq 0$ (or both), so the algorithm terminates.

UNIQNAME (print): ____________________

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` $\to$ `0`, `B` $\to$ `1`, …, `S` $\to$ `18`, `T` $\to$ `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 $0 \le i \le n$, 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 $d[i]$.**

   Solution: We have $d[0] = d[1] = 1$ since there is a unique way to decode the empty string, and a unique way to decode a single digit.
   For $i \ge 2$, we have that

   $$d[i] = \begin{cases} d[i-1] + d[i-2] & \text{if } C[i-1] = \texttt{1} \\ d[i-1] & \text{otherwise.} \end{cases}$$

   For the case $C[i-1] = 1$, we can either decode the $(i-1)$st and ith digits together as a single letter (from `K`–`T`), with $d[i-2]$ ways to decode the preceding digits; or we can decode the ith digit (as a single letter from `A`–`J`) separately from the rest, with $d[i-1]$ ways to decode the preceding digits. These two cases are disjoint and cover all decodings of $C[1,\ldots,i]$, so their sum is the total number of decodings.
   For the case $C[i-1] \neq 1$, we cannot decode the $(i-1)$st and ith digits together, because there is no corresponding letter; we must decode the ith digit separately, with $d[i-1]$ ways to decode the preceding digits.

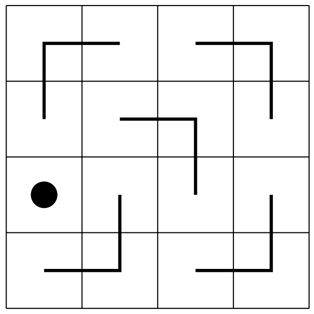
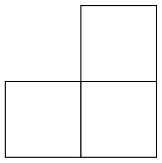
   (b) **Fill in the following table for the string C = `1110117101`.** Your answer will be graded for correctness based on the definition of $d[i]$, regardless of your recurrence.

i	0	1	2	3	4	5	6	7	8	9	10
$C[i]$	–	1	1	1	0	1	1	7	1	0	1
$d[i]$	1	1	2	3	5	5	10	15	15	30	30

UNIQNAME (print): ______________________________

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.)

      Solution: Part #1 is: place a tile so that the designated square is uncovered (and the other three are covered).

      Part #2 is: in the *two-by-two subgrid* at the *center* of the grid, choose the three squares from the three quadrants that do *not* contain the already-designated square.

      Part #3 is: Recursively tile each quadrant, each with its one designated square to leave uncovered.

(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.

Solution: For the base case, there is a two-by-two grid with one unit square to leave uncovered, and the algorithm covers the other three squares, as needed.

For the recursive case, by the choice of three additional squares to leave uncovered according to (PART #2), we are left with four $n/2$-by-$n/2$ quadrants, each of which has exactly one square to leave uncovered. By recursion/induction, the algorithm tiles those quadrants correctly, leaving the four designated unit squares uncovered. Finally, by the choice of the three additional squares in (PART #2), they are covered by the one tile as placed by the algorithm. This leaves just the original designated unit square uncovered, as desired.

(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**.

Solution: The recurrence is $T(n) = 4T(n/2) + O(1)$. This is because choosing (and finally covering) the three squares takes $O(1)$ time, since we just need to identify which quadrant the input square belongs to, and we make four recursive calls on subgrids of side length $n/2$.

This recurrence is covered by the Master Theorem, with $k = 4$, $b = 2$, and $d = 0$ (because $n^0 = 1$). Since $k > b^d$, the solution is $T(n) = O(n^{\log_b k}) = O(n^2)$.

All of the above bound are asymptotically tight, because choosing and covering the three squares also takes $\Omega(1)$ and hence $\Theta(1)$ time, and Master Theorem gives a tight solution in this case.

UNIQNAME (print): ______________________________

This page is intentionally left blank for scratch work.

If you have answers on this page, write “answer continues on page 13” in the corresponding solution box.

UNIQNAME (print): ______________________________

This page is intentionally left blank for scratch work.

If you have answers on this page, write “answer continues on page 14” in the corresponding solution box.