Problem 1

Solution by Anonymous Student

1. A Latin traversal for this square would be this. Each number of the traversal is noted with a cycle around it.

8 disjoint traversals in this case:
Generalization:
Let’s define some term.
Let $M(i, i+1)$ be a $2 \times 2$ block of the form

\[
\begin{array}{cc}
i & i+1 \\
+1 & i
\end{array}
\]

This is how we construct the first Latin square.
We will construct the first Latin square of the size $4m \times 4m$ from $4m^2$ $2 \times 2$ squares, each square $M(0,1)$, $M(2,3)$, ..., $M(4m-2, 4m-4)$ appears $2m$ times.

For example, for $m = 3$, the first Latin square would look like this:

\[
\begin{array}{cccc}
M(0,1) & M(2,3) & \ldots & M(4m-4, 4m-3) \\
M(2,3) & M(4,5) & \ldots & M(4m-2, 4m-1) \\
\ldots & \ldots & \ldots & \ldots \\
M(4m-2, 4m-1) & M(0,1) & \ldots & M(4m-6, 4m-5)
\end{array}
\]

We divide this $4m \times 4m$ square into four $2m \times 2m$ squares and call them A, B, C, D (as in picture number 2)
Each of the square A, B, C, D contains exactly $m^2$ $2 \times 2$ blocks of form $M(i, i+1)$.
For square A, we choose $m$ $2 \times 2$ blocks from that square such that no two $2 \times 2$ blocks are in the same row/column. Then we choose $m$ $2 \times 2$ blocks from square B such that the positions of those $2 \times 2$ blocks in square B are symmetric by the vertical axis to the square. Then we choose $m$ $2 \times 2$ blocks from square D such that the positions of those $2 \times 2$ blocks in square D are exactly the same as the positions of those $2 \times 2$ blocks in square A. Similarly, we get $m$ $2 \times 2$ blocks from square C.
For each transversal, we choose a number from a $2 \times 2$ block in square A/B and choose the other number from the corresponding $2 \times 2$ block in square D/C.
There are 4 ways to choose a number from a $2 \times 2$ square. There are $m$ ways to choose $m$ blocks
Thus, there are $4m$ ways to choose tranversal using this method. Since none of these tranversals use the same cell, they are disjoint.
So we can construct orthogonal pairs of Latin squares using this method.
Problem 2
Solution by David Aeschlimann

One can fit four dominoes onto the region in the problem statement and the shaded vertex cover in Figure 1 serves as a certificate that this solution is optimal.

![Figure 1: Adjacency matrix G](image)

Let $G$ be the adjacency graph of the region presented in the problem statement. Note that each edge corresponds to a possible location for a domino, and a matching on this graph corresponds to a possible solution of non-overlapping domino placements. The question then becomes: What is the maximum matching on the graph in Figure 1? By König’s Theorem, for any bipartite graph, the maximum size of a matching is the minimum possible size of a vertex cover. Figure 1 is a bipartite graph because there are no odd cycles, thus König’s Theorem applies. Further, the shaded vertices in Figure 1 represent a minimum vertex cover of $G$. This vertex cover is minimal by simple inspection, noting that you need at least four vertices to cover each of the four edges sticking out from the central square. Therefore, the maximum size of a matching in $G$ is four, which corresponds to the maximum number of dominoes that one can fit into the region specified in the problem statement.

Problem 3
Solution by Travis Chen (with some additions by Ben)

First as a base case we know that this is true for a graph with exactly 0 edges because each vertex is technically a chain with 1 vertex and 0 edges, and its degree is even. Next assume, take an arbitrary graph with $n$ edges, assume that every graph with all degrees even of lesser number of edges is a union of edge-disjoint cycles.

As a lemma, note that there must be at least one cycle in the graph. A graph without cycles is a tree, and this can’t happen because the leaves of a tree have odd degree. Alternatively, you can show there is a cycle using the same techniques used to prove Euler’s Theorem. Create a path by start at a vertex $v_0$ that has more than 1 edge, and travel to some neighbor $v_2$. Every vertex has even degree so we may leave $v_2$ along an edge that has not been used yet in the path. We repeat this until we return to $v_0$, which must happen because the graph is finite (The problem is not true for infinite graphs, e.g. an infinite line graph).

Go back to our arbitrary graph and remove the edges of this cycle that must exist. For every vertex of this cycle, we’ve removed exactly 2 edges so every vertex of this modified graph must also have even degree. By our inductive assumption, this modified graph must be reducible into a set of edge-disjoint cycles, so our original graph can be decomposed as this set plus the cycle we removed, completing the inductive step.
Problem 4

Solution by Clara Fannjiang

By construction. We color the vertices with a set of \( d + 1 \) colors according to the following greedy algorithm:

1. Sort the vertices by ascending label \( r(v) \).
2. For each vertex \( v \in V \) in sorted order:
   2.1. Color \( v \) with any color that none of its neighbors have already been colored with.

The algorithm iterates through every vertex, so every vertex is colored. We now prove that for each vertex in step 2.1, there is always at least one color out of the \( d + 1 \) colors that has not already been used by a neighbor. Since we iterate through the vertices in order of ascending label \( r(v) \), the only neighbors \( w \in N(v) \) that have been colored are those with \( r(w) \leq r(v) \). Since, by assumption, there are at most \( d \) such neighbors, there is at least one color out of the \( d + 1 \) colors that has not been used.

Given that we can label vertices with numbers \( r(v) \) such that for each vertex \( v \), at most \( d \) neighbors \( w \in N(v) \) have \( r(w) \leq r(v) \), we have constructed a proper vertex coloring with at most \( d + 1 \) colors. Therefore, the chromatic number is at most \( d + 1 \).

We note that applying this algorithm to the numeric labeling given by \( r(v) = 0 \) results in the trivial upper bound \( \Delta(G) + 1 \) mentioned in class, where \( \Delta(G) \) gives the maximum degree of the graph \( G \).