1. [10 points] Consider undirected graphs with multi-edges allowed, and self-loops not allowed. In class we proved that a graph has an eulerian tour if and only if it is connected (except possibly for isolated vertices) and every vertex has even degree. In this question we will consider what we can say about graphs that have a certain number of odd-degree vertices.

   (a) [4 points] Let $G = (V, E)$ be a graph on n vertices. Show that the number of vertices of G that have odd degree must be even. (There is a simple proof of this fact.)
   We begin by observing the following:

   $$\sum_{v \in V} degree(v) = 2|E|$$

   In other words, each edge is counted exactly twice when we sum the degrees of all the vertices. Let V_o be the set of odd-degree vertices, and V_e be the set of even-degree vertices. Now,

   $$\sum_{v \in V} degree(v) = \sum_{v \in V_o} degree(v) + \sum_{v \in V_e} degree(v) = 2|E|$$

   $$\sum_{v \in V_o} degree(v) = 2|E| - \sum_{v \in V_e} degree(v)$$

   Both terms in the righthand side of the second equation are even ($2|E|$ is even by definition, and each term in the summation is the degree of a vertex with even degree). In order for $\sum_{v \in V_o} degree(v)$ to be even, we must have an even number of terms (since each term in the summation is odd).
   Therefore, there must be an even number of odd-degree vertices.

   (b) [6 points] Now suppose G is connected and has exactly $2c$ vertices of odd degree. (We know from part (9a) that this number must be even.) Prove that it is possible to find exactly c paths (that can go through the same vertex more than once) that together cover all of the edges of G exactly once (i.e., each edge of G occurs in exactly one of the c paths, and that path contains the edge only once).

   Answer 1: We will show that it is possible to find d edge-disjoint walks (i.e., no pair of walks share an edge) that together cover all the edges in G by proposing a procedure to find them. We would like to make use of Euler's theorem but we cannot because of the existence of odd-degree vertices. Instead, we modify the approach that was used in class and in the notes to find an Eulerian tour. Suppose we start at any odd-degree vertex u, and walk along edges of G without repeating an edge. We claim that we must get stuck at some other odd-degree vertex v (which is different from u). This follows from a similar argument to the one we used for Eulerian tours: we can't get stuck at any even-degree vertex (because each time the walk visits a vertex we use two–an even number–of its edges); and we can't get stuck at u either because if we did then its degree would have to be even. So we get a walk from u to some other odd-degree vertex, v, that covers some of the edges of G. We now start from some other odd-degree vertex u' and walk from it, avoiding any edges previously used (including those on the first path). For the same reason as above, we must again get stuck at some other odd-degree vertex v' (and v' is not the same as either u or v, since they now have even degree after the edges

of the first path, from u to v, have been removed from consideration). This gives us a second walk, from u' to v'. Proceeding in the same way, we get a total of d edge-disjoint walks each connecting one odd-degree vertex to another. Finally, we may still not have covered all the edges of G. In that case, just as in the Eulerian tour construction, we start at any vertex w on one of our walks that is adjacent to an unused edge and walk along unused edges; by the usual argument, we must get stuck at w so we get a cycle. We then splice this cycle into the path (leading to a longer path with the same endpoints). We repeat this operation until all edges are covered.

Answer 2: Here is an alternative solution based on a clever trick. We add "virtual" edges in such a way that there are only even-degree vertices in the new graph. Then, we find an Eulerian tour in the new graph (by Euler's theorem, one exists). Once a tour is found, we remove the virtual edges and are left with d disjoint walks. More precisely, here is how the procedure works:

(i) Identify all the odd-degree vertices and pair them up in arbitrary fashion, say $\{u_1, v_1\}, \ldots, \{u_d, v_d\}$.

(ii) Construct a new graph G' by adding the d edges $\{u_i, v_i\}$ to G.

(iii) Since one edge has been added to every odd-degree vertex of G, all vertices in G' are of even degree. Thus, we can apply Euler's theorem (and run an appropriate algorithm) to find an Eulerian tour.

(iv) Finally, remove the d added edges from the tour. This will have the effect of breaking the tour into d segments, each of which is a walk (not a cycle). Moreover, the endpoints of the walks must be the odd-degree vertices of G (since these endpoints can only be at those places where we removed an edge). And the walks cover all the edges of G because the tour covered all the edges of G' and we removed only the virtual edges.

2. **Caution, Content is Graphic in Nature**[25 points] A question we might be interested in is if, given some number n, we can construct a single binary sequence containing each of the binary n-tuples exactly once. (Pretty interesting, right?) For example, with $n = 3$ we have 01110100, which contains 011, 111, 110, 101, 010, 100, 000, and 001 if we imagine the end of the sequence looping to the beginning. (For this question, you might be interested to know that a directed graph has an eulerian tour iff every vertex has the same in-degree as its out-degree, and the graph is *strongly* connected, that is you can reach any vertex from another vertex. You may use this fact without proof.)

   (a) [15 points] For the case $n = 3$ use a directed graph with nodes labeled 00, 01, 10, and 11, and explain why it makes sense that we can construct such a sequence. (Hint: For such a graph to be useful to us, what values and directions should we associate to each edge?)

   (b) [10 points] Using the method from part a, give a binary sequence that contains each binary string of length 4 exactly once.

   At the end you'll find a picture of sample graphs. The thing to note, is that every node can be left either by a 1 or 0, and every node can be reached either by removing a 1 or a 0 from the beginning. Hence, every node will be of degree 4. In general, no matter the node string

length, the nodes will have an out degree of 2, and an in degree of 2, for a total even degree, and allowing an Eulerian Tour to generate a tour over every edge (n-let) exactly once.

3. **You Can Always Count on Good Ol' Fermat**[25 points]
We're going to reprove Fermat's Little Theorem, from a counting perspective! (Obviously we aren't allowed to *use* the theorem below, then!)

   (a) [6 points] Show that $\binom{p}{k} = 0 \pmod{p}$ for $0 < k < p$, where p is prime.
   $\binom{p}{k} = \frac{p!}{(p-k)!k!}$ Notice that there is a factor of p in the numerator, and moreover there is no factor of p in the numerator (we don't want to divide by zero!) Hence, the entire fraction, $\binom{p}{k} = 0$.

   (b) [6 points] Show that $(x+y)^p = x^p + y^p \mod p$. It might help to use the **Binomial Theorem**: $(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$, where $\binom{n}{k} = \frac{n!}{(n-k)!k!}$.
   $(x+y)^p = \sum_{k=0}^{p} \binom{p}{k} x^{p-k} y^k$ Notice the first term will have $\binom{p}{0} = 1$ and $\binom{p}{p} = 1$ and by part a the terms in the middle will all have factors of p, and thus reduce to zero.

   (c) [13 points] Show that $a^p = a \pmod{p}$, by starting with the equality $0^p = 0 \pmod{p}$ and then using induction on a.
   We have seen the base case is true, now assume that $n^p = n \pmod{p}$, and now we want to show $(n+1)^p = n+1 \pmod{p}$, which is equivalent to $(n+1)^p = n^p + 1^p \mod p$ be b), and by hypothesis $n^p \cong n$ hence $(n+1)^p = n^p + 1^p = n+1 \mod p$. So we are done.

4. This problem counts more than the rest

   (a) How many 13-bit strings are there that contain exactly 5 ones?
   This is the number of ways to choose a subset of size 5 out of a set of size 13 (that is, the choice of which five positions in the string will contain a 1). Thus, the answer is $\binom{13}{5}$.

   (b) How many 55-bit strings are there that contain more ones than zeros?
   The number of 55 bit strings which contain more ones than zeros is equal to the number of 55 bit string which contain more zeros than ones (since, for example, we can pair these two sets up in a bijective correspondence by pairing each string with the result of flipping all its bits). Furthermore, these two distinct sets together comprise all the 55 bit strings (in particular, there are no 55 bit strings which contain the same number of ones and zeros, as 55 is odd). Thus, the number of 55 bit strings which contain more ones than zeros is precisely half the total number of 55 bit strings, and thus equal to $\frac{1}{2} \times 2^{55} = 2^{54}$.

   (c) How many different 13-card bridge hands are there? (A bridge hand is obtained by selecting 13 cards from a standard 52-card deck. The order of the cards within a bridge hand is considered irrelevant.)
   This is the number of ways to choose a subset of size 13 out of a set of size 52, and therefore equal to $\binom{52}{13}$.

(d) How many different 13-card bridge hands contain no aces? [Recall that there are four aces in a standard 52-card deck]

This is the number of ways to choose a subset of size 13 out of a set of size 52 - 4 = 48 (the 48 non-ace cards), and therefore equal to $\binom{48}{13}$.

(e) How many different 13-card bridge hands contain all four aces?

Such a hand is determined by its 13 - 4 = 9 non-ace cards, which can be any of the 48 non-ace cards in the deck. Thus, there are $\binom{48}{9}$ such hands.

(f) How many different 13-card bridge hands contain exactly 5 spades?

Such a hand is a combination of 5 spades and 13 - 5 = 8 non-spades. The 5 spades are selected from the 13 spade cards in the deck. The 8 non-spades are selected from the 39 non-spade cards in the deck. Thus, there are $\binom{13}{5} \times \binom{39}{8}$ such hands.

(g) How many ways are there to order a standard 52-card deck?

There are 52 choices for the first card in the ordering, then 52 - 1 choices for the second card in the ordering, then 52 - 2 choices for the third card in the ordering, and so on, till ultimately there is only one choice left for the last card. Thus, there are $52 \times 51 \times 50 \times ... \times 1 = 52!$ many ways to do this.

(h) How many different anagrams of "KENTUCKY" are there?

In general, by the same reasoning as above, the number of anagrams of a word is equal to the factorial of its length, divided by the product, for each character which appears multiple indistinguishable times within that word, of the factorial of the number of times that character appears.

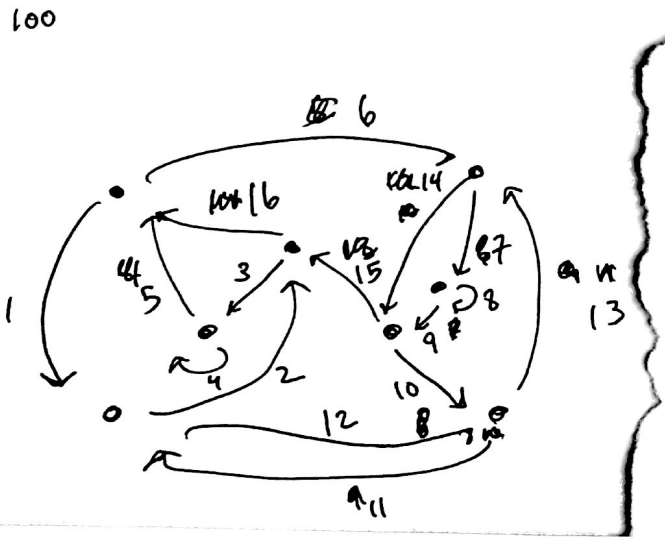
Thus, in this case, the answer is $\frac{8!}{2!}$ (8 letters total, 2 of which are both K).

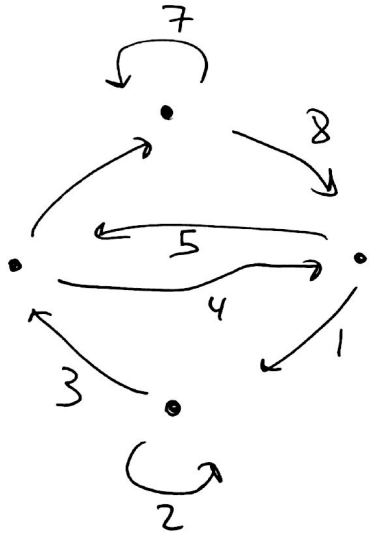
(i) How many different anagrams of "ALASKA" are there?

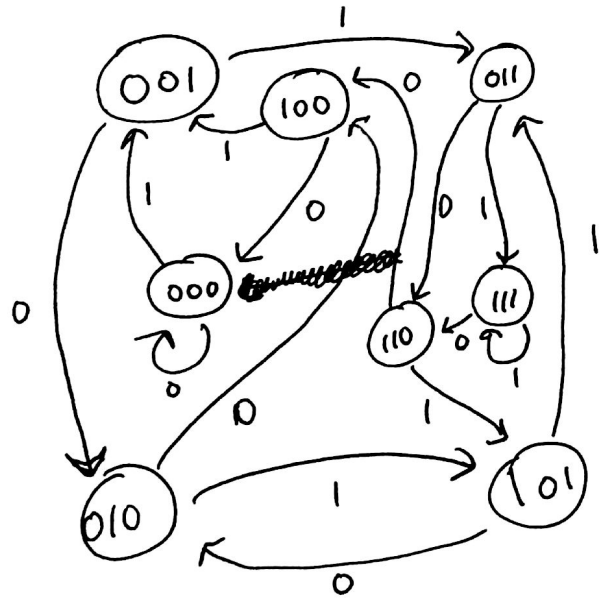
By the reasoning above, this is $\frac{6!}{3!}$ (6 letters total, 3 of which are A).

(j) How many different anagrams of "MISSISSIPPI" are there?

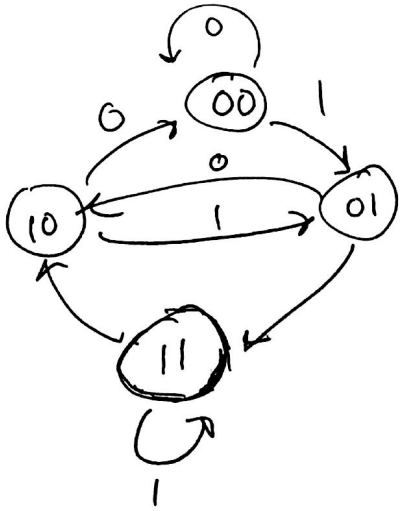
By the reasoning above, this is $\frac{11!}{4!4!2!}$ (11 letters total, 4 of which are I, 4 of which are S, and 2 of which are P).







00100001111010111001



01110100