I collaborated with:

Problem
An Euler[1] tour of a graph G is a circuit (that is, a path that begins and ends at the same vertex), through G that traverses every edge of G exactly once.

(a) Prove that a connected graph G has an Euler tour if and only if every vertex has even degree.

(b) Describe and analyze a linear time algorithm to compute an Euler tour in a given graph, or correctly report that no such tour exists.

Solution
a) Assume, for the sake of contradiction, that a vertex v in a connected graph G that has an Euler tour has odd degree. Once we visit v through an edge, we need to leave it through another edge, meaning that visiting and leaving v takes two edges. Since v has odd degree, we will encounter a situation that we have visited all but one edges incident to v. At this point, we visit the last unvisited edge and end up in v. Now, we do not have any other unvisited edges so we would have to use a visited edge to leave v. We arrive at a contradiction, so if G has an Euler tour, then every vertex of G has even degree.
Let G be a connected graph such that all of its vertices have even degree. Now we must proof that G has an Euler circut because of this. We can proof this by strong induction:
Base Case: Let N be a connected graph containing zero edges. This implies that there is only one node in N as a node is connected to itself. We notice N has an Euler tour as we can visit this node by starting our tour from it.
Inductive Hypothesis: Assume that a graph of i edges, where $0 \leq i \leq k$ for some integer k, contains an Euler tour, with the condition that each vertex in the graph has even degree.
Inductive Step: We must now show that a graph with $k+1$ edges where all vertices have even degree contains an Euler tour. Let G be such connected graph with $k+1$ edges with all its vertices having even degree. We know that G contains a circut, because its vertices have even degrees and we can always travel back to a vertex. Let us remove a circuit c from G. By doing so we now have subgraphs of G with each having less than $k+1$ edges. From our inductive hypothesis we know these subgraphs contain Euler tours. We also know these subgraphs are all connected to c because if they were not, then our graph would not be connected. We can start a tour by picking a vertex of c and walking around it until we reach said vertex again. When we reach one of the subgraphs we can do an Euler tour on it and come back from the node where we entered it. We do this for all subgraphs connecting to c until we reached the vertex where we started our tour from. We walked through every edge of c as well as every edge of every subgraph exactly one. We also visited every vertex of G. Thus, G contains an Euler circuit because it has vertices of even degrees.
b) Algorithm in page below.
If our graph has a vertex with odd degree, then our prework will be able to identify this and return an error. Now, suppose there is an edge e in G that is unvisited. If we did not encounter e in the first while loop, when we look at the circuit containing s, then e must be in a subgraph. However, we also look at all the ciruits in the subgraphs. We also know that there will keep existing circuits because every vertex has even degree and when we look for the circuits we always remove two edges at a time, the one going into a vertex and the one going out. So if it is not in the first circuit we walked through nor in any cuircuit in any subgraph, then e is not in G. We have reached a contradiction as e cannot be both in and not in G. Thus, this algorithm finds an Euler tour.
Runtime: We go over every edge randomly once. We walk through this path again, going over those edges we had not visited twice. $O(|V| + |E|)$
Space Requirement: relative to number of edges $O(|E|)$

[1]Named after Leonhard Euler (1707-1783) who solved part (a) in 1735

Algorithm 1 Find Euler Tour

```
$C$ is our sequence containting the Euler tour.
$s$ is our starting vertex
Prework:
for every vertex $i$ in our graph do
    if $i$ has odd degree then
        return error.
    end if
end for
function EULER($G$,$C$,$s$)
    Walk through an edge incident of $s$
    while current vertex $v \neq s$ do
        remove edge used to get to $v$
        move through another indicent edge
    end while
    Begin walking down same path as before
    add $s$ to $C$
    while current vertex $u \neq s$ do
        add $u$ to $C$
        if $degree(u) \neq 0$ then
            $P \leftarrow$ subgraph containing $u$
            $Euler(P, C, u)$
        end if
    end while
    add $s$ to $C$
    return $C$
end function
```