Drew Goldman
Wiring - Description

The Wiring problem requires a unique implementation of Kruskal's algorithm. The objective of the problem was to correctly wire a house based on a list of copious restraints, as listed in the problem description, the most notable of which is the limiting of edges in the creation of the Minimum Spanning Tree. I approached this problem by creating Graph, Edge, and Node classes to encapsulate the input data. Essentially, the algorithm added all the nodes that were given as input to an unordered map along with its unique integer identifier to help distinguish the nodes. Adding the nodes requires $\Theta(V)$ time. The map was a great choice because it allows for constant time lookup of the nodes when traversing the edges. Then the algorithm traversed through the edges and added ONLY the edges that were valid, that is, edges that did not violate any of the restraints of simply some node to another node. The method valEdge() confirmed this; however, valEdge() did not check for all possible invalid edges, as some of the verification needs to occur within the Kruskal's portion. Adding the edges requires $\Theta(E)$ time. Finally, the main method calls the Kruskal's algorithm. The only major modification to the Kruskal algorithm, to create the correct MST, was checking whether a switch had already been connected to a box, outlet, or breaker. This functionality was implemented in the switchConditional() method. If a switch had already been connected to a box, outlet, or breaker, then a switch cannot be connected to another box, outlet or breaker; therefore, Kruskal's does not add an edge if that is the case; otherwise, Kruskal's will add that switch edge. Interestingly, switchConditional() is $\Theta(V)$ because it iterates through the vector results to check among all the added nodes. Since switchConditional is called within each Kruskal's iteration, which is itself $\Theta(E*\log(V))$ due to the efficiency added from find() and Union(), the **total runtime of this algorithm is $\Theta(E*V*\log(V))$**.

No pseudocode required, as the code is available to the grader via Gradescope.