Final Report:

This program (final_reattempt.rs) creates and analyzes a graph with edges and nodes. The graph, sourced from https://snap.stanford.edu/data/ego-Facebook.html, describes circles (friend list networks) on Facebook. The dataset has node features, or profile information, circles, and ego networks (networks from the perspective of one node or person (the ego)). All the features are anonymized and just described as feature_1 and so on. For this data, the edges are undirected. The NodeID is the individual person. The .edges are those in their ego network. The .circles contain a list of NodeIDs. The .feat files are for each of the nodes, and .egofeat is for the individual nodes. .featnames has booleans for 'has/does not have feature'.

In this project, I am investigating the connectivity and isolation of subgraphs within the overall network. To do this, I calculated the five most influential nodes in the entire network (by degree), and then found the most isolated subgraphs (finding the threshold for number of components was an iterative process, but the threshold seems to be around 500). I then found the most influential node of that subgraph and calculated the path distance between the most influential whole-network node and the most influential isolated network node. I returned this as well as the average path length for all paths from the whole-network node, and the average path length for the isolated-network node. This information allowed me to compare exactly how isolated these isolates were, and draw conclusions about the dynamics of the entire network based on that.

To achieve that, my code was structured as follows:

-Before main(), I created line_reader and average_path. These are fairly straightforward and self-explanatory; line_reader reads data, and average_path calculates the average path length. Average_path uses Dijkstra's algorithm, the standard tool for finding path lengths. It calculates the lengths and then averages them.

-Within main(), there are no args and its return type is simply io::<Result>.

-The first function is graph construction, where I use ungraph (because the edges are undirected).

-I then identify the 5 most influential nodes by degree and store them.

-I then partition the graph with connected components. I used UnionFind to identify disjoint sets in the whole network. Then I create a hashmap to store each component (disjoint set) with its member nodes.

-I use that information to create isolated subgraphs, networks where the number of member nodes is less than 500. In theory, these subgraphs represent unconnected networks.

-I then calculate the path lengths and compare them.

My results were interesting. Here is the output for a run on one of the several available sets:

Distance from entire-network node NodeIndex(33) to subgraph node NodeIndex(33): 0
Average path length from entire-network node NodeIndex(33): 1.75
Average path length within subgraph node NodeIndex(33): 1.75

Distance from entire-network node NodeIndex(15) to subgraph node NodeIndex(33): 1
Average path length from entire-network node NodeIndex(15): 1.82
Average path length within subgraph node NodeIndex(33): 1.75

Distance from entire-network node NodeIndex(7) to subgraph node NodeIndex(33): 1
Average path length from entire-network node NodeIndex(7): 1.84
Average path length within subgraph node NodeIndex(33): 1.75

Distance from entire-network node NodeIndex(91) to subgraph node NodeIndex(33): 1
Average path length from entire-network node NodeIndex(91): 1.87
Average path length within subgraph node NodeIndex(33): 1.75

Distance from entire-network node NodeIndex(48) to subgraph node NodeIndex(33): 1
Average path length from entire-network node NodeIndex(48): 1.86
Average path length within subgraph node NodeIndex(33): 1.75

These results were repeated throughout every file and through multiple thresholds. Essentially, the most influential node is also the most influential subgraph node, and the distance from the other influential nodes to the subgraph node is always 1. This indicates that there is no actual isolated subgraph, and the entire network is extremely well-connected. This is supported by the fact that the average path lengths never exceed 2, indicating that any node can reach any other in at most 2 steps.

This represents efficient information flow, small-world characteristics (tightly knit and interconnected) and high connectivity in general. These conclusions make sense, given that this is facebook data.