Project 1 Part A: Pseudocode
Dr. Russel Pears

Jessie Li
Hiram Dominguez

1. i=0
   Set k as the budget
   WHILE clock ticks is less than or equal to clock ticks limit
      IF data collection reaches 10 hrs from 8am to 6pm THEN
         10*60*60 = 36,000 clock ticks
      END IF
   Generate a bidirectional graph with 60 locations randomly
   Randomly assign the length of a road to each edge of the graph
   Update the matrix with new generated trips

2. Compute benefits matrix Benefit(X,Y)
      The benefits matrix will list out the nodes with the longest distance first
      Calculate Benefit (X, Y)
         Benefit(X,Y) = $spd(X,Y) - d(X,Y) * (nt(X,Y) + nt(Y,X))$

         $+ \sum_{n_1 \epsilon N(Y)} Max(spd(X, n_1) - d(X,Y) - d(Y, n_1), 0) * nt(X, n_1)$

         $+ \sum_{n_2 \epsilon N(X)} Max((spd(Y, n_2) - d(Y,X) - d(X, n_2)), 0) * nt(Y, n_2)$

3. Sort matrix in descending order
4. Suppose that L, M is the entry that has the highest benefit
      WHILE i < k
         Find the entry with the highest benefit in the sorted matrix
            Build roads between locations L and M
               Increment I by 1
      Determine how the road will affect the adjacent nodes.
   Update the matrix by the new road between L and M
5. Recommend that roads L and M be built
      IF Dijkstra's algorithm is less than the original distance, THEN
         build the road
6. i=i+1
7. Update all entries in the matrix that are affected by roads L, and M being built
      Calculate benefits for the roads that connect L and M
      Adjust the benefits matrix

8. If (i<k) go to step 3 else return

Initialize variables

Clock ticks =equals 0

N: number of locations (nodes) in the network

P: average connectivity of nodes across the network

L: road length parameter between 5 to 25

T: number of trips generated at each clock tick

K: number of new roads to be budgeted

Shrinkage factor (f) is equal to 0.8

Generate random number of trips

Randomly select start location

Randomly select end location

//use Dijkstra's algorithm to determine the shortest path length between the start and end locations

FUNCTION Dijkstra (Graph, Source):

FOR each vertex (V) in Graph(G)

Distance[V] <- INFINITY

Previous[V]<- NULL

Set distance [source] <-0

WHILE queue(Q) IS NOT empty

U<=node in queue with smallest distance[]

Remove U from queue

FOR each unvisited neighbor of vertex of U

temp <- distance[U] + distance_between(U,vertex)

IF temp <-distance[V]

distance[V] <-temp

previous[V] <-U

END IF

return distance[], previous[]

https://www.programiz.com/dsa/dijkstra-algorithm

Graph =

Benefit (0,2)=(19-19*0.6)*(1+1)

Benefit (0,3)=(16-16*0.6)*(1+1)

Benefit (1,2)=(25-25*0.6)*(1+2)

Benefit (1,4)=(15-15*0.6)*(2+1)+(25-(9+10))*(1+2)