Group Member Names:

Vincent Lee, Gustavo Necochea Aguayo, Emmett Lim, Andrew Arsenault

Algorithm 2 Pseudocode:

```
/* NOTES:
city_distance ~ city i is city_distance[i] away from city i + 1
fuel ~ fuel[i] is equal to the gas available at city i
mpg ~ number of miles the car can travel per gallon of fuel
Can only traverse through cities clockwise, last city connected to first city
Gas tank always starts off empty, then gets filled on starting city based on the given fuel there
Assume there is always at least one optimal starting city
*/

func best_starting_city(city_distance, fuel, mpg):

    // 1st for loop for every new trip
    for(size_t i = 0; i < city_distance.size(); ++i):
        gas_tank = 0 // Stored in gallons
        starting_city = -1 // optimal starting city
        city_n = 0 // tracker for current city index

        // 2nd for loop for actual trip loop
        for(size_t j = 0; j < city_distance.size(); ++j):
            <add current city fuel into gas_tank>
            <set city_n equal to (i + j) % city_distance.size()>
            // note: using city_n as current city stopped at
            <calculate the total fuel left (gas_tank) from current city to next city>

        if gas_tank is negative: <get out of 2nd loop to start in next city>
        else if gas_tank >= 0 and did full city loop:
            <set starting_city to index>
```

if starting_city positive value:
<get out of 1st loop as optimal path is found>
return index of opitmal path (starting_city)

Proving Efficiency for Pseudocode:

Proving Efficiency for Pseudocode using Limits:

Step 1: Label Variables

T(2n^2) = Function to represent the logic of Algorithm 2
f(n^2) = informed guess about efficiency class of Algorithm 2

Step 2: Fill in Variables

T(n) = 2n^2
f(n) = n^2

Step 3: Prove T(n) ε f(n) using limits

$$\lim_{n\to\infty} T(n)/f(n) = \lim_{n\to\infty} 2n^2/n^2$$

$$\lim_{n\to\infty} T(n)/f(n) = 2$$

Which is non negative and constant with respect to n. Therefore 2n^2 ε O(n)

Proving Efficiency using Step Count

Outer Loop: O(n) Time efficiency

Runs n times starting at city 0 and iterates through all cities. Needed to calculate the distance for the best starting city.

Inner Loop: O(n) Time Efficiency

Also runs n times for each iteration of the outer loop. Calculate gas available at each city.

Total Step -> T(n) = n * n == n^2
Therefore Time Complexity = n^2