A0258448N Ngoh Pek Suan, Nikki
A0252779M Adam Seah Jun Hui
A0252754B Jackson Yeong Hong Han
A0254813E Seck Miao Zhen Christabel

1. Explain the purpose of the algorithm in your app. Clarify whether the chosen algorithm is the only existing algorithm that can perform the intended task. List alternative algorithms and explain your choice.

The purpose of our algorithm in our app is to recommend users the shortest and most eco-friendly route based on factors such as distance they will be travelling, mode of transport and the amount of carbon emissions for each route option. The algorithm will be using a database to calculate the carbon emissions for each route and ranking it. This algorithm is not the only algorithm that can perform the intended task. An alternative algorithm would incorporate PageRank algorithm where it assigns a 'PageRank' score to each route based on other user's selection history and ranks and recommends it based on PageRank score. For example, if many people select a certain route option, it will be bumped up the ranks on the route recommendation page.

2. Explain how this algorithm works - what are the inputs and outputs, what is the sequence of steps performed on this input and leading to the output (you can illustrate this with a flow chart)?

Step 1: The program begins by retrieving the current database stored as .csv files on Google Colab. These files contain vehicle emissions data and are processed to construct a nested dictionary for computation. While ideally, we would integrate the Google Maps API function directly, for demonstration purposes, we have created a mock-up.
Step 2: Using the function `random.random()`, the program generates an arbitrary floating-point value, rounded to two decimal places, within the range of 0.0 to 50.0. This value represents the distance for later computation when the user inputs their start and end points.
Step 3: A function is created to calculate the estimated travel time for different vehicle types based on the 2 locations input by the user divided by the speed of the vehicle.
Step 4: This step involves defining the user's profile based on their car model, weekly CO2 quota, and remaining CO2 emissions for the week so that we can calculate the user's total and remaining emissions in step 10.
Step 5: This section is where the user interaction begins. The program prompts users to input their start and end locations. The function `google_maps_distance_km()` calculates the distance based on the provided inputs by the user.
Step 6: Another function, `emissions_per_trip`, is defined to compute the total carbon emissions for a given journey duration and vehicle model used.
Step 7: A nested dictionary is created to store the time taken and emissions for each vehicle type to travel a given distance. This categorization includes Car, Public Transport, Walking, and Bicycle. For Walking and Biking, emissions are assumed to be zero since they don't rely on fuel. However, for Car and Public Transport, because they rely on fuel (unless it is a pure electric vehicle). For Public Transport, a specific consideration is made: it's assumed that if users choose this mode of transport, half the distance will be covered by MRT and the other half by Bus transport. Therefore the program divides the total distance by 2. Each half is then individually processed to calculate the time taken and carbon emissions using the function google_maps_time_taken(). These values are rounded and added together to obtain the cumulative time and emissions for the entire journey via Public Transport.
Step 8: A ranking algorithm is implemented. The nested dictionary from Step 7 is converted into a sortable variable type, enabling sorting. The Python `sorted()` function is used to arrange the routes in

descending order by emissions and time taken. However, the algorithm also considers the practicality of each route. If a route significantly exceeds the travel time for the fastest route for 1 hour or more, it will be ranked lower, even if it's more eco-friendly.
Step 9: The program displays the ranked routes to the user, based on the sorting performed in Step 8.
Step 10: User input is required to select a preferred route. Once chosen, the program deducts the carbon emissions from the user's remaining CO2 quota for the week and provides feedback on the remaining emissions. This corresponds to the prototype's sliding bar interface, indicating the user's remaining carbon emissions for the week.

3. What is (are) the limitations of this algorithm?

Dummy values
For the purpose of this assignment, we assigned arbitrary dummy values for data such as the CO2 emissions of each mode of transportation. Therefore, it may not give the algorithm realistic datasets to process. While our current plan does not involve the algorithm to perform machine learning, a more complex algorithm in the future that can make predictions about a user's future carbon emissions will require supervised learning. Using dummy values for this will prevent us from verifying the accuracy of the algorithm's predictions with the 'ground truth'.

Inaccessibility of Google Maps API
As Google is a for-profit company, Google charges for access to its Google Maps API. Thus, this prevents us from prototyping the actual product at a low cost. The non-public nature of Google Maps API can also be due to security reasons to prevent unauthorised illicit usage of Google Maps API.

Dummy codes
As a result of our inability to access Google Maps API for free, our code therefore can be considered as 'dummy code' as it generates a random distance for a user's journey for demonstrative purposes. Thus, it differs from the actual code needed to connect to Google Maps API and retrieve distance information. With dummy codes, we will not be able to verify the actual feasibility of the product's ability to retrieve Google Maps API data in a real-world scenario.

Sorting algorithm unable to completely empathise with needs/preferences of users
The sorting algorithm will not be able to fully account for the needs and preferences of individual users with regards to the distances which they are willing/able to walk/cycle. The sorting algorithm currently simply demotes slower modes of transportation (usually walking/cycling) to the bottom of the list when they are more than 1 hour longer than the fastest option. However, this simply assumes that everyone's threshold is this constant 1 hour, whereas some people may consider not walking if it is longer than 15 mins or some other lower threshold.

Some users may have characteristics that entail special considerations, which the sorting algorithm will not be able to empathise with. For instance, a person with mobility issues may not be able to access all routes suggested, as some pedestrian routes are not accessible to wheelchairs. A person's inability to walk longer distances, such as the elderly, is also not accounted for. Situational factors which may affect the needs of users are not considered. For instance, a user without an umbrella when it rains will require sheltered paths and transfers between all sectors of the journey, which the product does not account for.