ASSIGNMENT 3
Spectrum Based Fault Localization

ANSHUL PUROHIT(231110007)

IMPLEMENTATION:

1. FITNESS SCORE

   I'm using DDU(Density-Diversity-Uniqueness) Metric to get the fitness score.

   - For calculating Density, I'm computing the density of the activity matrix, which is the ratio of the sum of values to the total number of values in the matrix.
   - To calculate Uniqueness, I'm counting the occurrences of each row in the activity matrix using the Counter class from the collections module and calculating the sum of (count * (count - 1)) for each unique activity.
   - For calculating Diversity, I'm doing the following task:
     - If there is only one row in the activity matrix, diversity remains 1.
     - Otherwise, it's calculated as 1 - similarity_sum / (len(activity_mat) * (len(activity_mat) - 1)), which is a measure of the uniqueness of the rows in the matrix.

   After normalizing the Density, I'm using the DDU metric formula to generate the fitness score.

2. SUSPICIOUSNESS

   Here I'm using the Ochiai Metric to calculate the suspiciousness of a particular component.

   C_f : Denotes the number of failing tests that execute C

   C_p: Denotes the number of passing tests that execute C

   N_f: Denotes the number of failing tests that do not execute C

   N_p: Denotes the number of passing tests that do not execute C

A loop iterates through the errorVec and column_data to compare their values. The loop distinguishes four cases:

- If errorVec[i] is 0 (indicating a non-error) and column_data[i] is 0, it increments n_p (non-error predicted as non-error).
- If errorVec[i] is 0 and column_data[i] is 1, it increments c_p (non-error predicted as error).
- If errorVec[i] is 1 (indicating an error) and column_data[i] is 0, it increments n_f (error predicted as non-error).
- If errorVec[i] is 1 and column_data[i] is 1, it increments c_f (error predicted as error).

After getting all the values and solving it will give the suspiciousness of a particular component.

3. getRankList
   I'm calculating and ranking components based on their suspiciousness scores. It sorts the components in descending order of suspiciousness and assigns ranks based on the order.

ASSUMPTIONS:

I'm assuming that I'm getting a good set of test suite because activity matrics is dependent on it. And the program should be somewhat similar with fewer bugs.

LIMITATIONS:

I'm limiting the line of code to somewhat less than 100 because otherwise, it'll take a lot of time if there are multiple branches in the program.