Assignment 3 Report

Name: Akash Shivaji Varude

Roll Number: 231110006

1. Implementation Details:
   - **Fitness function:**
     I calculated Ulysis score for finding fitness of activity_matrix.

$$\mathcal{L}_i = \begin{cases} c_j \mid c_j \in C, j \neq i, & \text{if } c_i = \vec{0} \\ c_j \mid c_j \in C, c_j = c_i, j \neq i, & \text{otherwise.} \end{cases}$$

$$\mathcal{W}_i = \frac{|\mathcal{L}_i|}{m-1}$$

$$\mathcal{W}_{Ulysis} = \frac{1}{m} \sum_{i=1}^{m} \mathcal{W}_i$$

     To calculate |Li|, I just traversed activity matrix columnwise and count of columns exactly similar to component ci. Using this, I calculated Wi for each component and stored it in the list W.
     To get W_ulysis, I just calculated average of all of the elements of list W.

   - **Suspiciousness function**
     I used ochiai score for calculating suspiciousness of a component
     Cf: Number of failing tests that execute component
     Cp: Number of passing tests that execute component
     Nf: Number of failing tests that do not execute component

     Initially I set Cf, Cp, Nf all to zero
     We already have component vector and error vector also.
     Now I traversed through all the values of component and error vector

Component value at index i	Value of error vector at index i	Operation
1	1	Cf+=1
1	0	Cp+=1
0	1	Nf+=1

     after calculating all these values, I calculated final ochiai score using the formula

$$Ochiai(C) = \frac{C_f}{\sqrt{(C_f+N_f).(C_f+C_p)}}$$

- **GetRankList Function:**
  Let we have [component, suspiciousness_score] in descending order
  e.g.
  `[['c1', 0.70], ['c2', 0.70], ['c3', 0.66], ['c10', 0.66], ['c5', 0.57], ['c6', 0.57], ['c9', 0.57], ['c4', 0.5], ['c8', 0.5], ['c7', 0.0], ['c11', 0.0]]`
  Now i have to make ranklist as
  `[['c1', 2], ['c2', 2], ['c3', 4], ['c10', 4], ['c5', 7], ['c6', 7], ['c9', 7], ['c4', 9], ['c8', 9], ['c7', 11], ['c11', 11]]`

  Explanation:
  Here c1 and c2 have same score hence assigned them rank 2.
  c3 and c10 have same score hence assign them rank 2+2 i.e. 4
  why 4? answer-> two ranks already used. Now there are two components with same score hence assign them rank 4
  c5, c6, c9 will have rank=4+3=7 i.e. four ranks already used and there are three components with equal score hence 4+3=7
  similarly ranks for other components are also calculated.

  Pseudocode I used :

  i. For each component of activity matrix
     a. Calculate suspiciousness score by calling suspiciousness function
     b. Store the component and it's score in list as [component, score] in 'component_score' list
  ii. Sort the component_score' in descending order of score (named it 'Sorted_Components')
  iii. Calculate how many components have same score and store it in 'same_score_id_count' list
     Explanation:
     In above given example, 'same_score_id_count' list will look like
     `[['c1', 2], ['c2', 2], ['c3', 2], ['c10', 2], ['c5', 3], ['c6', 3], ['c9', 3], ['c4', 2], ['c8', 2], ['c7', 2], ['c11', 2]]`
     This is because component c1 and c2 i.e. only two components have same score 0.70. Hence `['c1', 2], ['c2', 2]`
     Component c3 and c10 i.e. only two components have same score 0.66. Hence `['c3', 2], ['c10', 2]`
     Component c5, c6 and c9 i.e. only three components have same score 0.5. Hence `['c5', 3], ['c6', 3], ['c9', 3]` and so on.

  iv. `rankList[0]=` same_score_id_count[0]
  v. `Travers i from 1 to m    //m is number of components`
     a. `if(sorted_components[i][1]!=sorted_components[i-1][1]):`
        `rankList[i] = [same_score_id_counts[i][0],rankList[i-1][1]+same_score_id_counts[i][1]]`
     b. `else:`
        `rankList[i]= [same_score_id_counts[i][0],rankList[i-1][1]]`

Assumption and Limitaiton:

Evolutionary Search Based Test-suite Generation algorithm given in the assignment can work if there is only 1 error in buggy program in one path

IR of a turtle program is consider as components.

if len(IR) = 5 then components are 0, 1, 2, 3, 4 i.e c_0, c_1, c_2, c_3 and c_4

Sample Output:

```
Optimized Activity Matrix:
 [[1 1 1 1 0 0 1 1 0 1 0]
 [1 1 1 0 0 0 0 0 1 1 0]
 [1 1 0 0 0 0 0 0 0 0 1]
 [1 1 1 1 1 1 0 1 0 1 0]]

error Vector:
 [0 1 0 1]

components with their Ochiai scores sorted in descending order:
 [['c0', 0.7071067811865475], ['c1', 0.7071067811865475], ['c2', 0.6666666666666666], ['c9', 0.6666666666666666], ['c4', 0.5773502691
896258], ['c5', 0.577350269189625 8], ['c8', 0.577350269189625 8], ['c3', 0.5], ['c7', 0.5], ['c6', 0.0], ['c10', 0.0]]

final ranklist:
 [['c0', 2], ['c1', 2], ['c2', 4], ['c9', 4], ['c4', 7], ['c5', 7], ['c8', 7], ['c3', 9], ['c7', 9], ['c6', 11], ['c10', 11]]

DONE..
```