Consider the following program and its mutated version in which we replace the multiplication operator (*) with the sum operator (+) at line 9:

```
1 public static double[] multiply(int[][] matrix, int[] vector) {
2    int rows = matrix.length;
3    int columns = matrix[0].length;

4    double[] result = new double[rows];

5    for (int row = 0; row < rows; row++) {
6        double sum = 0;
7        for (int column = 0; column < columns; column++) {
8            sum += matrix[row][column]
9                    * vector[column];
        }
10        result[row] = sum;
    }
11    return result;
  }
```

```
1 public static double[] mutatedMultiply(int[][] matrix, int[] vector) {
2    int rows = matrix.length;
3    int columns = matrix[0].length;

4    double[] result = new double[rows];

5    for (int row = 0; row < rows; row++) {
6        double sum = 0;
7        for (int column = 0; column < columns; column++) {
8            sum += matrix[row][column]
9                    + vector[column];
        }
10        result[row] = sum;
    }
11    return result;
  }
```

Consider the following test case:

```
int [][] matrix = new int[3][3];
matrix[0][0] = 2
matrix[0][1] = 0
matrix[0][2] = 2
matrix[1][0] = 2
matrix[1][1] = 1
matrix[1][2] = 3
matrix[2][0] = 0
matrix[2][1] = 2
matrix[2][2] = 4

int[] vector = new int[3];
vector[0] = 2
```

```
vector[1] = 0
vector[2] = 2
```

Perform weak mutation analysis and specify at which point of the execution the mutant is marked as dead and why (consider single statements as segments).

The answer should

1. indicate the sequence of statements up to the statement at which the execution kills the mutant. The sequence of statements should be a sequence of statement-numbers that refer to the code, for example, write 1 2 3 4 to indicate the execution of the first four statements of the program.

2. explain why the mutant is killed by indicating the difference in the states of the original program and the mutant that allows for killing the mutant

The original program will produce the following output:

[4.0, 8.0, 4.0]

The mutated program will produce the following output:

[6.0, 4.0, 6.0]

The sequence of statements up to the statement at which the execution kills the mutant is:

1 2 3 4 5 6 7 8 9

The difference in the states of the programs becomes apparent at statement 9. This difference leads to different calculations, resulting in different outputs and causing the mutant to be killed by the provided test case.