Implementation:

I have define checkeq() function inside symbSubmission.py file.

#Step1

In which First I have Load the json Data of both the testdata(testData1- A complete program without any unknown constant parameters, testData2- A program with some unknown constant parameters.)

#Step2

Then code begins by iterating through the keys of testData1 and testData2. This means it will consider each entry/testcase in both datasets, and then we compare the parameters(using 'params') of json mean we are comparing the inputs of both the files if we got a match then we make dictionaries for parameters/inputs , and for symbEnc(which contain the corresponding outputs)

#Step3

Then we add all the symbols/variables by using addSymbVar() function.

#Step4

Then we take output for each symbol/variable from both the testData and then concatenate them by using "==" then we add them as constraints in z3 solver. we do this for all variables.

Then z3 solver take all the constraints and if all will satisfy then it will give the value of unknown parameters/constants.

Five Interesting Test Cases:

1. eqtest1.tl(with constant parameter)-testData1.json
   eqtest2.tl (complete program)-testData2.json

```
sat
[c2 = 22, c1 = 18]
-------
```

2. Example1_p1.tl(with constant parameter)-testData4.json
   Example1_p2.tl (complete program)-testData3.json

```
sat
[c2 = 25, c1 = 15]
-------
```

3. Example2_p1.tl(with constant parameter)-testData5.json
   Example2_p2.tl (complete program)-testData6.json

```
sat
[c2 = 83, c1 = 40]
-------
```

4. Example3_p2.tl(with constant parameter)-testData8.json
   Example3_p1.tl (complete program)-testData7.json

```
sat
[c2 = 50, c1 = 40]
-------
```

5. Example4_p1.tl(with constant parameter)-testData11.json
   Example4_p2.tl (complete program)-testData12.json

```
sat
[c4 = 40, c3 = 30, c2 = 20, c1 = 10]
-------
```

Assumption:

The code assumes the existence of two JSON files,which are opened for reading and parsing. These JSON files likely contain data related to test cases.

We are assuming that while generating the testData.json we are giving the same initial input in both programs.

testData.json for p1.tl

python ./kachua.py -t 100 -se ./example/p1.tl -d '{\":x\": 5, \":y\": 100}' -c '{\":c1\": 1,\":c2\": 1}'

testData.json for p2.tl

python ./kachua.py -t 100 -se ./example/p2.tl -d '{\":x\": 5, \":y\": 100}' -c '{\":c1\": 1,\":c2\": 1}'

in both the commands inputs(x,y) has same initial inputs (5,100)

Limitation:

1. Code will not give correct result for unknown constants if we give the unknown constants inside conditional statements like:
   If :c1>25 [
   …
   …
   ]
   Here for this type of program code will not give correct result.
2. Code will give sat(satisfiable) only when both programs has same structure. If structure both the program are different then code may give incorrect result.
3. The code generates constraints based on the "symbEnc" values of matching test cases. However, the code does not perform any validation or error handling to ensure that the "symbEnc" values are valid expressions. Invalid expressions could lead to errors or incorrect constraint generation.

4. If while testData generation if user will not give same initial input then program may give incorrect result.