Some pairs of test cases are created for the purpose of checking different cases.

1. **test_1 and test_2 :**

   Both the files are identical but at one point it is different. test_2 contains one else statement but test_1 doesn't. But they are representing the same. So, in both cases if no other conditions are fulfilled then the program will return 5 and for test_2 the return 0 is redundant.

   As per the logic, they are showing equivalent in our tool.

2. **test_3 and test_4 :**

   In these files, the order of the conditions are changed keeping functionality semantically the same. Moreover, these programs contain complex conditioning with "&&". But our tool is able to handle this situation also and results in equivalence.

3. **test_5 and test_6 :**

   These files basically contain a simple function of finding the maximum integer out of two integers. There are two ways to express this conditioning (Either compare y>=x or x>=y). Though the variable is different in the two cases, they have successfully passed the test in our tool.

4. **test_7 and test_8 :**

   One condition like x<100 can also be written as x<=99. So, in these test cases, all the conditions are written in a very different way. Though they are syntactically very different, semantically they are identical. Our tool is also successful in finding the total equivalence between them.

5. **test_9 and test_10 :**

   The order of conditions present in test_9 is changed in test_10 and only one variable constraint is present in those programs. When we run it on our tool, they are showing as non-equivalent which should not be the case as they are functionally equivalent. The limitation reflected in this case is that the same variable is symbolically represented in a different way. So, if we change the variable name manually in the constraints_1 or in constraints_2 file, Then the command "python runLogicalSummary.py constraints_1.txt constraints_2.txt" would output the equivalence.

6. **test_11 and test_12 :**

   It is almost the same as the test_3 and test_4 but the order of conditions conjuncted by "&&" is changed. But our tool is successful in finding the equivalence in them.

7. **test_13 and test_14 :**

   These ones are with much more complex conditions containing both "&&" and "||" and with the change of orders. This test was also successful.

   Conversion of test source to binary:

gcc <test_1.c> -o <test_1>

This command is used to generate all the binaries present in the testBinary folder. If the file name is test_1.c in the testSource file, the corresponding binary file would be test_1. This convention is followed for all the examples.