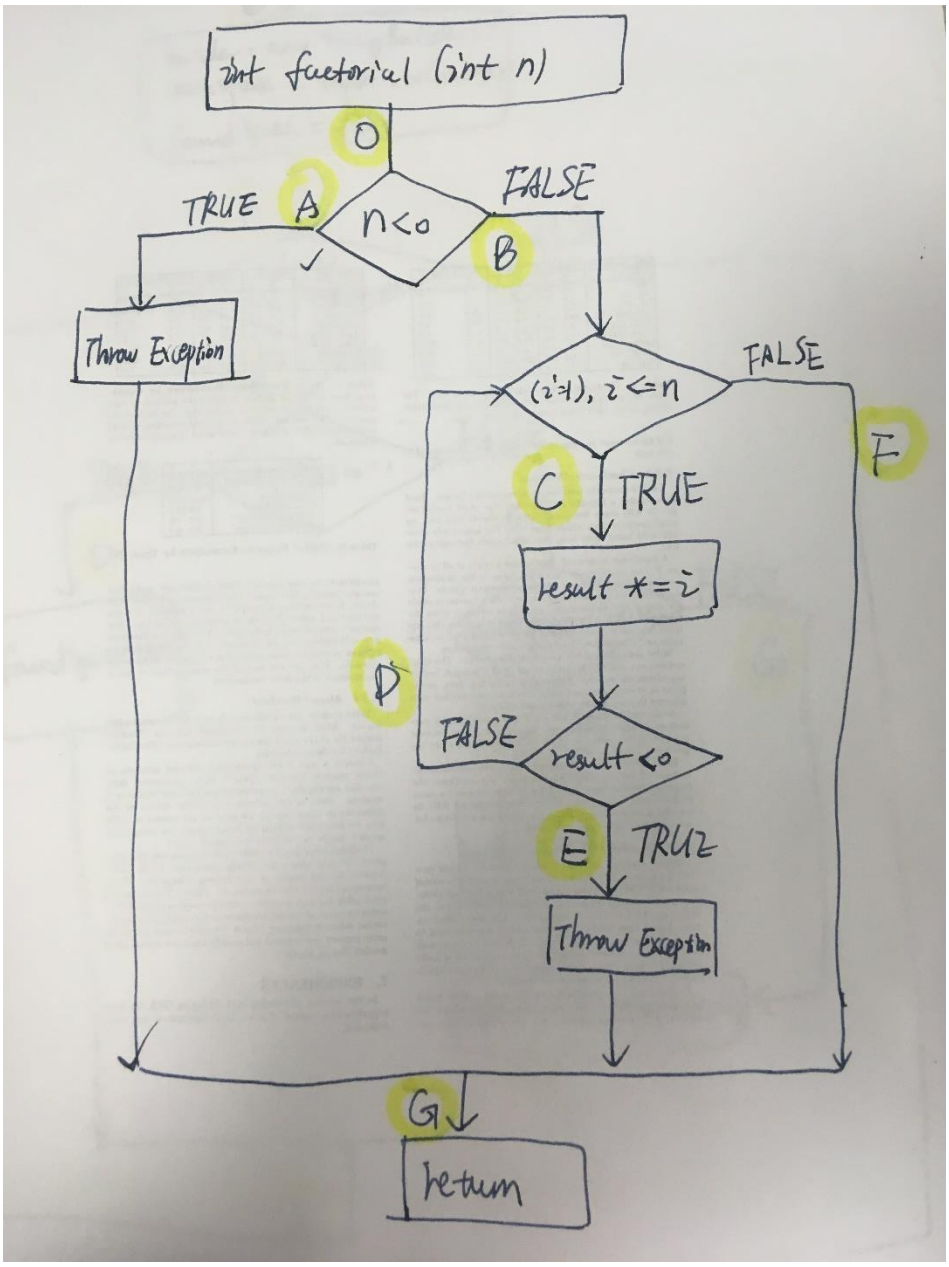
Problem 1.

1. Control Flow Graph



2. Test Specification for branch coverage

As we know that, branch coverage is that we design testcases to make sure that for every possible branch from each decision point is executed at least once, ensuring that all the reachable code is executed.

From the Control Flow Graph above, there are totally 4 paths in the CFG, they are:

- OAG
- OBFG
- OBCDFG
- OBCEG

Now, we could specify the following test cases to get 100% percent coverage.

Test cases No.	n	Result	Path
1	-1	IllegalArgumentException	OAG
2	1	1	OBCDFG / OBFG
3	13	RuntimeException	OBCEG

3. Is this test spec sufficient to make a decision?

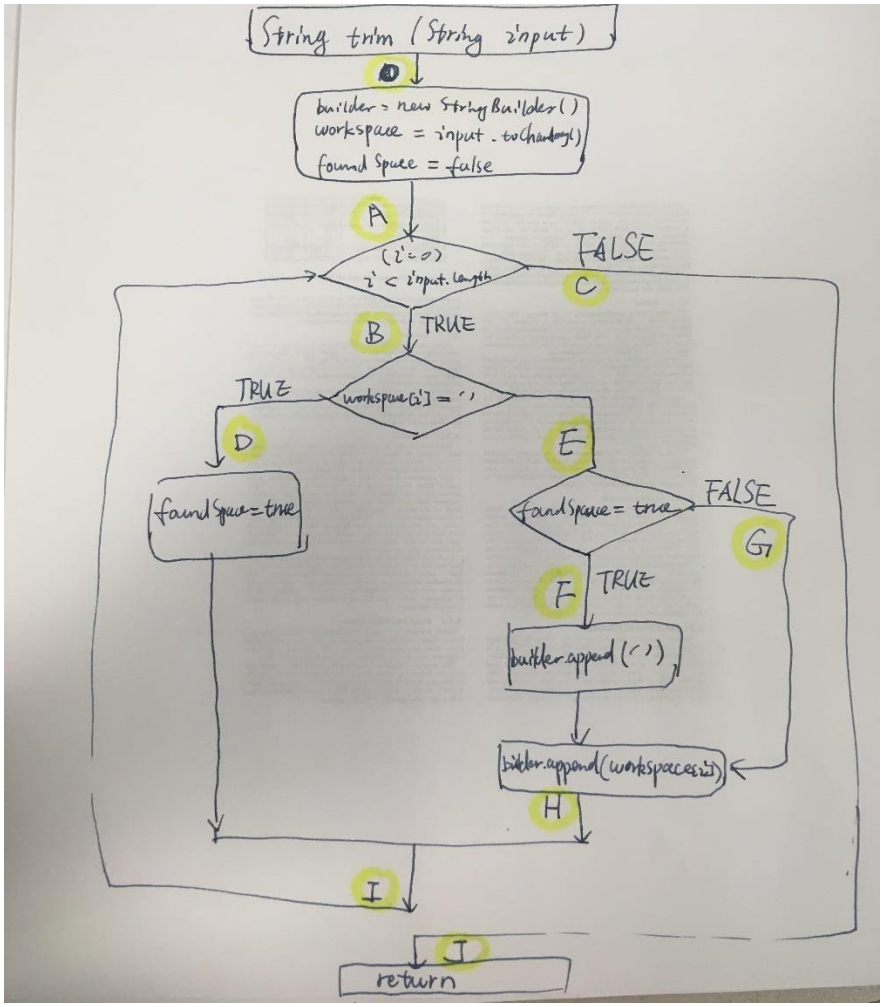
No, I think it is insufficient. Although the spec has 100% branch coverage, which means that it goes through every branch from every decision point at least once. For the input n, if n < 0, it will go through the OAG path, the this is covered by the test case 1. If n is between 1 and 12 inclusively, it goes through the OABCDFG path, which is covered by test 2. If n is greater than 12, it goes through OBCEG, which is covered by test case 3. The four cases do not cover all the input range, it misses n = 0, which should go through path OBFG directly, and return 1 in the end. So, it is insufficient.

To make it sufficient, an extra test should be added:

Test cases No.	n	Result	Path
4	0	0	OBFG

Problem 2

1. Control Flow Graph:



2. Test Specification for branch coverage

From the Control Flow Graph above, there are totally 4 paths in the CFG, they are:

- OACJ
- OABDICJ
- OABEGHICJ
- OABEFHICJ

Now, we could specify the following test cases to get 100% percent coverage.

Test cases No.	input	Result	Path
1	"an apple"	"an apple"	All 4 paths

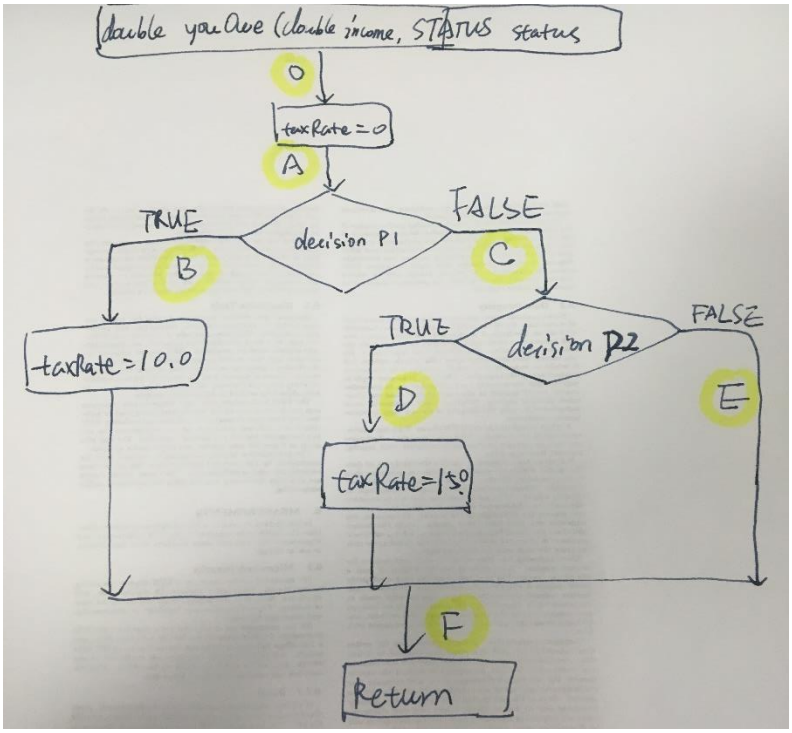
3. Is this test spec sufficient to make a decision?

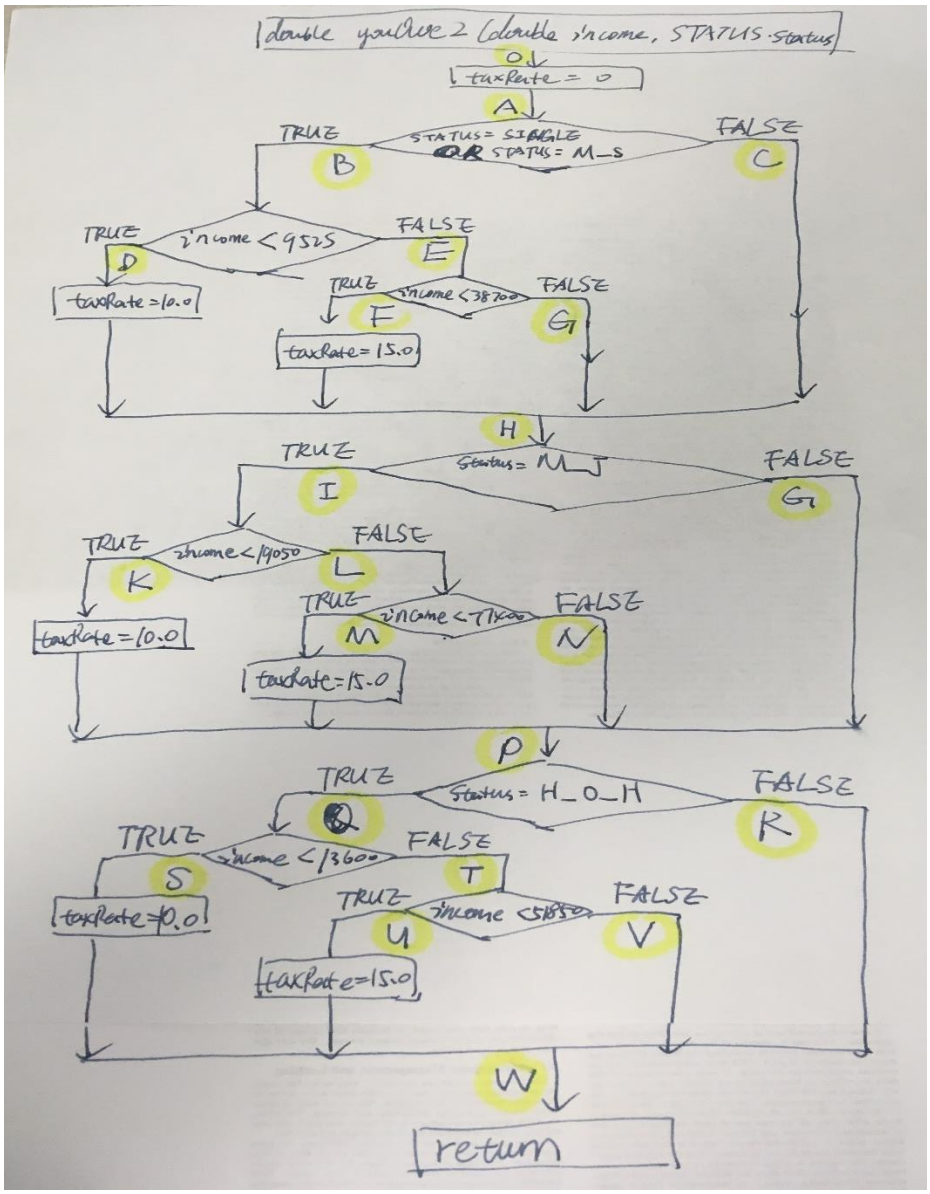
No, it is insufficient. The test case 1 has a 100% coverage rate, because it goes through every branch from every decision point for at least once. While it misses some cases that go through the 4 paths in different ways. For example, it misses input = "", which goes through path OACJ directly, and it misses input = " ", which goes through just one path that is OABDICJ. Besides, it misses input = "apple", which goes through OABEGHICJ. Even for OABEFHICJ, it misses the cases for continuous spaces, like input = "an apple". All of these make it insufficient to make a decision.

Problem 3

From structural testing perspective, one should take the version2. The reasons are as follows:

1) Although the Flow Control Graph of version1 is simple, it has much complicated decision point than version2. While for version2, every decision point has one or two conditions, and it is clear for us to see how many statements, branches and conditions we have in the program, which make easier for us to come out tests cases that could cover every branch and every condition.





2) From 1) version1 can simply get a 100% statement coverage, it needs only 2 test cases to get there, with a testcase of first decision point being true which go through path OAB and the other first decision being false and second decision point being true which go through path OACDF, it will cover all the statement in the method. Also, it can simply get 100% branch coverage, too. It needs only 3 test cases to cover, they are tests cases covers these 3 paths in the program, OAB, OACDF and OACEF. All these only covers a little portion of all the combinations for each decision point.
However, for version2, we could easily figure out all the statements, branches and conditions. There three main parts in its CFG, A to H, H to P and P to W. And we could come out with test cases that covers every conditions of each parts. For example, for the first part, OABDH, OABEFH, OABEGH, OACH, and for second part, HIKP, HILMP, HILNP, HGP, and for the third part, PQSW, PQTUW, PQTVW, PRW. We know that there could be only one fit among the 3 decision point A, H and P, so we could narrow the testcases down to 12, 4 for each. While for decision point A, it has 2 conditions, have to double the test cases for STATUS being SINGLE and MARRIED_SPERATELY. All these leaves us 16 test cases to cover.

So in a word, we prefer version2 over version1.