CS 743 Assignment 1

```
/*
* @param the method has four parameters: int pressure, int volume, int velocity, boolean
lowFuel
* @return a String about execution status
*/
public String importantFunction(int pressure, int volume, int velocity, boolean lowFuel ) {

... }
```

1 . For the above funcEon, use ACTS tool to generate test cases when t=2 and when t=3.

- pressure enum 5, 10, 15
- volume enum 200, 300, 400
- velocity enum 1, 2, 3, 4, 5
- lowFeul Boolean true, false 2. Given the funcEon below:

```
        public String importantFuncEon(int pressure, int volume, int velocity, boolean lowFuel )
        { if (pressure < 10) {

        if (volume > 300) {
        if (velocity == 5) {

        System.out.println(" A bug happened! ");

        return " A bug happened! "; }

        }
        else if (lowFuel) {

System.out.println(" Everything looks good! "); return " Everything looks good! ";

}
}
else {

return " Everything looks good! "; }

return null; }
```

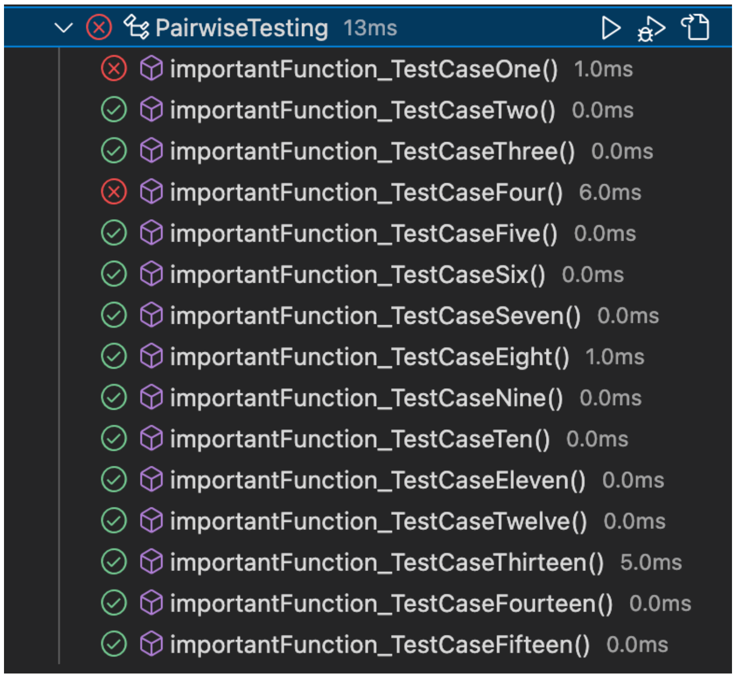
Using JUnit in Eclipse to write your JUnit test cases for t = 2. Execute your Junit test program.

```
System.out.println(" Everything looks good! ");
```

3. How to do automate testing? What are the difficulties?

Automated testing can be difficult because there are several parts that need to work together. It was easy to create automated tests for the example code by using the test cases generated by ACTS and implementing them with Junit. This is because I could easily hardcode the test values for each of the fifteen cases before running them. However, the number of test cases quickly increase as the t-value increments, so I would not want to hardcode each testcase. For this reason, it would be ideal to pull test data from a text document provided by ACTS and loop through each line of the file, testing the cases while going. This way, the testing suite would be more flexible and less time consuming to setup. However, I don't know enough about Junit because I couldn't get the program to recognize the path of text files. I think it is executing from a directory that is different from my project directory. Additionally, I would use the Assertions.assertAll() method, so each test case would run from the loop before terminating if the test failed.

This is just one example of automated testing and a very simple one at that. It becomes even more difficult as the project size grows and the scope of the testing increases. My problems with filepaths and directories seem minuscule when compared to some of the Automated UI and Web App testing projects I have worked with. Below are the results of my tests:



```
✗ PairwiseTesting 13ms
    ✗ importantFunction_TestCaseOne() 1.0ms
    ✓ importantFunction_TestCaseTwo() 0.0ms
    ✓ importantFunction_TestCaseThree() 0.0ms
    ✗ importantFunction_TestCaseFour() 6.0ms
    ✓ importantFunction_TestCaseFive() 0.0ms
    ✓ importantFunction_TestCaseSix() 0.0ms
    ✓ importantFunction_TestCaseSeven() 0.0ms
    ✓ importantFunction_TestCaseEight() 1.0ms
    ✓ importantFunction_TestCaseNine() 0.0ms
    ✓ importantFunction_TestCaseTen() 0.0ms
    ✓ importantFunction_TestCaseEleven() 0.0ms
    ✓ importantFunction_TestCaseTwelve() 0.0ms
    ✓ importantFunction_TestCaseThirteen() 5.0ms
    ✓ importantFunction_TestCaseFourteen() 0.0ms
    ✓ importantFunction_TestCaseFifteen() 0.0ms
```