Question 1

(a)

Three functional tests for this program are as follows:

Test 1:

Check for valid user input. This test will verify that the user inputs calculate the correct amount for the car insurance payment according to the formula used to calculate the customer's car insurance payments. The test will fail if the program fails to compile or returns a null/empty value or returns an incorrect car insurance payment value.

Example pseudocode:

```
def carInsurance()

age, yDriving, claimAmts = 0
postalCode = ''
payment = 0

try:
//formula to calculate insurance


except Exception:
    print("Incorrect value or null return val. Check input")
    sys.exit(1)

return 0
```

Test 2:

Check for invalid input. This test will verify that the program can correctly handle negative inputs or invalid input types. The expected output in such cases will be an error message or thrown exception.

Example pseudocode:

```
def carInsurance()

age = -100
yDriving = five
claimAmts = -1
postalCode = 239548
payment = 0

try:
//formula to calculate insurance
```

```
except Exception:
    print("Invalid input values.")
    sys.exit(1)

return 0
```

Test 3:

Check for edge cases. This test will check for some common test cases in such calculations, such as input values that are correct but don't result in a correct car insurance payment. For example:

- Age: 21
- Postal Code: 290 234
- Years of driving: 3
- Number of claims in the last 3 years: 0

This input values are valid, but the software test might not catch such an exception, which is why it's important to test for such edge cases.

(b)

Three boundary value tests for this software are as follows:

1. Test for a minimum and maximum age, this test will verify that input age values fall within a certain range and the program handles the calculation correctly within these bounds.
2. This test will verify that the program will correctly calculate care insurance payments based on a minimum and maximum years of driving in relation to the customer's age and number of claims. In doing so, this will account for some erroneous edge cases.
3. This test will verify that the program correctly handles the input number of claims within certain bounds in relation to the other input values (e.g., minimum 1 claim, maximum 15 claims).

(c)

Three exploratory tests that could test the veracity of this program's calculations are as follows:

1. Test for erroneous or unusual inputs, so inputs that are valid but result in an incorrect car insurance payment. For example, an input area code for a place that no longer exists or has been renovated, re-developed (new postal code) or moved. With such a value the program will return a value, but it won't be correct and such cases need to be accounted for.
2. Test for consistency between input values. For example, an input value of age 50 and years of driving 0 would not make any sense, or should at least escalate the customer's application to be investigated and verify their information and extenuating circumstances.

3. Test for invalid combinations of input values. The program needs to account for an incorrect combination of values, that while valid and result in a return value, are not correct and need to be accounted for. For example, an input age of 15. and number of claims 10, makes no sense and the program needs to account for such cases.

(d)

One way to improve this program's specification is to properly define the bounds of the input values and their restrictions in relation to the company's policies and state regulations. This can be achieved by stipulating the ways in which the algorithm should function to meet the company's needs, or by even providing an explicit formula that the company uses and how the input values relate to the formula.

These improvements will make it easier to check for the program's correctness, and it would potentially make it easier to iterate over, as strong fundamentals of the program will have been established and more focus can be put toward QOL improvements for both users and insurance agents.