Tu Lam

CS 362 / Instructor Vijay Tadimeti

February 4th, 2021

Homework #4

1. *Volume of a cube*

i)

Answer: The code for this program will be provided with the GitHub URL link to the file.

URL: https://github.com/tulam1/cs362-hw4/tree/main/volume%20cube

ii) *Explain your thinking behind those tests that you chose to write*

Answer: When writing the test of the volume of the cube, I must think of three cases that can happen during the run of the program. For the first test case, this is simple as we enter a number and see if the number will display correctly based on the result is calculated. The second case that I choose is to test a negative number and since my program check for negative number, it will never pass as the negative cannot continue with the program. Lastly, the last test case I went with a floating number, this will work but still return a failure as the placement of the decimal is not to the precision that we want and the case will fail and this is why it only accepts a whole integer number for the program.

2. *Average of elements in a list*

i)

Answer: The code for this program will be provided with the GitHub URL link to the file.

URL: https://github.com/tulam1/cs362-hw4/tree/main/average%20element

ii) *Explain your thinking behind those tests that you chose to write*

Answer: In the program of the average of elements in a list, the three cases I chose to pick to test out the unit test is simple. The first

is simple as I input in a list with some element and find the average and see if it prints out correctly. The next test case for the unit test is I give an empty list into the test case, and the case would return a failure as I assert as false if list pass in the parameter received a NULL list. Lastly, I test cases the divisible by 0, since no number can be divisible by it, the case would fail if it were detected. These are the three test cases I chose to test case against the unit test for this average element program.

3. *Generates a full name*

i)

Answer: The code for this program will be provided with the GitHub URL link to the file.

URL: https://github.com/tulam1/cs362-hw4/tree/main/full%20name

ii) *Explain your thinking behind those tests that you chose to write*

Answer: In the program for generating a full name by via input. I did three test unit cases that revolve around the input. The first case is simple is that the user provided the first and last name and it combine it together to produce the full name. The second case if each string is blank/empty, the case will fail as nothing is entered to produce the full name. Lastly, the last case resolves if either the first or last name is empty, then the case will fail, and nothing will display.

Link to GitHub w/ all 3 programs: https://github.com/tulam1/cs362-hw4

4. *From In-class activity 1*

i) *Choose a component from your system from In-class activity 1. Describe specifically what your component does in your system.*

Answer: Going back to the In-class activity 1 where would think of a system that a social media app would have, and I end up choosing Snapchat. In the system, there are a lot of components that make up the app, but in this part, I will focus on one component in the app which is the texting component. The texting component in Snapchat allows you to text to your friend, share video/photo, copy/paste, and more. That the basic of the component as it acts like a normal text message app in the system of Snapchat.

ii) *Write down three examples of unit tests you could write for that component.*

Answer: Below will give three examples of unit tests I could write for texting component.

1. I could do a test that would check on the limit on how word can hold in the text box. If the user enters below or equal to the word limit, the test will pass or else not, fail.
2. I could do a unit test on the component as the user sends photo/videos and check the limit if they send more than 20 photo/videos to a friend. If the number is below or equal to, it will pass the test. Else, the test will fail and ask user to limit down the sending.
3. I could do a unit test to test the word that user enter in the text box is depend on the language they type. If the language they type matches with the component, it will recommend word to finish your sentence in that language. If the word cannot be recognized in any languages, no recommendation is made on that word.