Quiz 03

Due Sep 27 at 10pm **Points** 10 **Questions** 5 **Time Limit** None

Instructions

Answer the following questions in your own words. Do NOT simply cut and paste the information from the slides. You will receive a score of 0 if you copy the prose from the slides.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	58 minutes	0 out of 10 *

* Some questions not yet graded

Score for this quiz: **0** out of 10 *
Submitted Sep 26 at 6:22pm
This attempt took 58 minutes.

Question 1

Not yet graded / 2 pts

Describe Test Driven Development. Which comes first, the code or the tests?

Your Answer:

Test Driven Development (TDD) is a process of software development that works on the repetition of a very short development cycle wherein the Requirements are turned into test cases and then the code is modified so that the tests pass.

The main objective of TDD is to make code clear and simple. It encourages the developer to write test cases for each functionality. If the automated test cases fail then the user needs to modify the code. This reduces the duplication of code.

In TDD, the **test comes first** then comes the code.

Question 2

Not yet graded / 2 pts

What are the advantages and disadvantages of debugging with print statements?

Your Answer:

Advantages of debugging with print statements are as follows:

1. It is easy to use. (Beginners can easily use them to debug their code.)
2. It is flexible. (You can add them anywhere without any restrictions.)
3. It becomes easy to see what is happening in the program while it running we can get an idea until which block the code is executing perfectly.

DisAdvantages of debugging with print statements are as follows:

1. It becomes tough to remove all print statements that have been used for debugging but are not necessary for running the actual program since you have to manually check the necessity of the statement.
2. This cannot evaluate other functions that we need to evaluate. Let's assume we wrote a print statement to check the value of A+B then we need to go back to the code then modify it and run it again to check the value of A-B whereas in debug we can directly put up the expression in terminal to check the value.
3. The readability & understandability becomes substantially less when we use print statements inside recursive functions.
4. We can not pause the execution in order to check the value of other variables and then continue or go step by step based on our own requirements.

Question 3

Not yet graded / 2 pts

How do breakpoints help with debugging?

Your Answer:

A breakpoint is an intentional stop or pausing point added in order to pause at the specific portion, analyze then continue based on the situation.

Breakpoints help us in debugging in the following ways:

- It gives you the flexibility to pause the execution of the code on any part.
- This helps in navigating through the execution of the code step by step.
- Once we set breakpoints, we can check the values of global and local variables while the code is executing.

Question 4

Not yet graded / 2 pts

What is the difference between step in and step over debugger commands?

Your Answer:

Step In: Step In basically executes the code line by line. If the line contains a function and we click on the step in then it will go inside the function line by line and show us the detailed line by line execution.

Step Over: Step Over basically steps over a given line of code. If the line contains a function and we click on the step over, then it executes the function directly without getting inside the function and executing line by line.

Question 5

Not yet graded / 2 pts

Describe the divide and conquer strategy for debugging.

Your Answer:

Divide and conquer is an iterative process of dividing the code into half until you find the bug. For example, if there are 500 lines of code in our program and we need to find out the bug in it, it becomes substantially difficult to go line by line through the code. In such kind of a scenario, we can use divide and conquer strategy wherein we will divide the code in half and then figure out which half of the code is not running as expected then again divide the part containing the bug into half and run the code. We need to iterate this process until we finally narrow down to the exact portion where the bug is located.

Quiz Score: **0** out of 10