**Knowledge and Theory: Testing**

**Part 1 – Introduction:**

**Testing a Real World Object:**

* Understand the Product and its purpose. Who will use it? How will they use it? Why will they use it? For example, a paper clip could be used to hold papers together by teachers or it could be used by artists to bend and unlock sim cards in cellphones. For the latter, you’ll want to test if it will break when you bend it straight.
* Gather all the use cases.
* What are the bounds of use? For example, how many papers should the paper clip be able to hold? How much bending should it be able to take? How long should it last?
* What are the stress faillure conditions? This is the maximum weight/quantity/load that it can handle before it begins to fail at doing what it is designed to do. For example, a laundry machine washer you could ask how much pieces of clothing should it be able to wash properly? If the users says at maximum 40 pieces, then you would have to test this faillure condition.

**Software Testing:**

* **Automated Vs. Manual testing:** You want to provide as much automated testing as possible. There are some components/features that are easier to test manually than it is to test using automation. For example, detecting if an image is unnacceptable (perhaps contains nudity) on a social network that was reported by a user.
* **Black Box Testing Vs. White Box Testing:** With Black Box Testing, we are given the software “as is” and are asked to test it. All we know is how it should work. We cannot look into the “Black Box”. In White Box Testing, we have programmatic access to test individual methods. It’s generally smarter to use automation for white box testing and manual for black box testing.

**Testing a Method:**

* Have accurate understanding of what the method “should” do**.** For example, if the method is an array sort, you should know how the output array should be organized.
* Test the normal case.
* Test the extreme or end cases. This is usually the empty, size one, or really large size. These are the cases that are valid but on the edges.
* Test the invalid case: Usually when the person entered an invalid paramter. Nulls, negative numbers, whatever is considered an invalid input.
* Test the strange cases: i.e. for an array sorting method, test the case where the array is already sorted. Any case that could potentially make the method trip out.
* Write unit tests that includes all of this.

**Troubleshooting:**

* **Undestand the Scenario:** For example, the client says “chrome is crashing”. You figure out that chrome version 3.0.1 is crashing for users on Windows 10.1, when they have 9 tabs already open and the open the 10th tab. An error report has already been logged.
* **Walk Through Use Case:** You run through the entire use case very slowly and realized that the menu pane runs of space and throws an exception when a new tab is entered.

**Question 12.2:** *You are given the source to an application which crashes when it is run. After running it ten times in a debugger, you find it never crashes in the same place. The application is single threaded, and uses only the C standard library. What programming errors could be causing this crash? How would you test each one?*

* There is a random variable that if contains a particular value, causes the bug. Look for any variables that have any random component to it.
* Look for un-initialized variables that could have any state during the call.
* Look for any memory leaks.
* Look for any external dependancies. It could be the state of some other entity that this application relies on that is having random behaviour.
* Try to figure out any pattern in which it crashes or doesn’t.

**Question 12.3 -** *We have the following method used in a chess game: boolean canMoveTofint x, int* *y). This method is part of the Piece class and returns whether or not the piece can* *move to position (x, y). Explain how you would test this method.*

* Invalid Cases: Test negative numbers for X and Y. Test X and Y values that are larger then the valid range.
* Edge Test: Test with a full board. Test with an empty board.
* Test with more black pieces. Test with more white pieces.
* Discuss with your interviewer whether you should return false or throw exceptions when tests fail.

**Question 12.4 -** *How would you load test a webpage without using any test tools?*

* Ask about response time, throughput (number of transactions per second), resource utilization (i.e. percent of CPU or memory) and maximum load the system can bear (i.e. maximum traffic) . Test all of these individually.
* We could write a Java multithreaded program (perhaps on different virtual machines) that will test all these automatically (and maybe spawn other threads to test traffic).

**Question 12.5 -** *How would you test a pen?*

* Ask who will be using the pen, what will it be used for, when will it be used, how will it be used, why will it be used.
* Ask about characteristics about this pen (i.e. should it wash off easily on clothes?). What about its size? Is it ball point? What is the color?
* Test all the two points above.