Software Testing Interview Questions And Answers Guide.

Global Guideline.

https://globalguideline.com/

Software Testing Job Interview Preparation Guide.

Question # 1

A good software tester should be Intelligence?

Answer:-

Intelligence.
Back in the 60's, there were many studies done to try to predict the ideal qualities for programmers. There was a shortage and we were dipping into other fields for trainees. The most infamous of these was IBM's programmers' Aptitude Test (PAT). Strangely enough, despite the fact the IBM later repudiated this test, it continues to be (ab)used as a benchmark for predicting programmer aptitude. What IBM learned with follow-on research is that the single most important quality for programmers is raw intelligence-good programmers are really smart people-and so are good testers.

Read More Answers.

Question # 2

A good software tester should be Skeptical?

Answer:-

Skeptical.
That doesn't mean hostile, though. I mean skepticism in the sense that nothing is taken for granted and that all is fit to be questioned. Only tangible evidence in documents, specifications, code, and test results matter. While they may patiently listen to the reassuring, comfortable words from the programmers ("Trust me. I know where the bugs are.")-and do it with a smile-they ignore all such in-substantive assurances.

Read More Answers.

Question # 3

low cunning. Street wise is another good descriptor?

Answer:-

"low cunning." "Street wise" is another good descriptor, as are insidious, devious, diabolical, fiendish, contriving, treacherous, wily, canny, and underhanded. Systematic test techniques such as syntax testing and automatic test generators have reduced the need for such cunning, but the need is still with us and undoubtedly always will be because it will never be possible to systematize all aspects of testing. There will always be room for that offbeat kind of thinking that will lead to a test case that exposes a really bad bug. But this can be taken to extremes and is certainly not a substitute for the use of systematic test techniques. The cunning comes into play after all the automatically generated "sadistic" tests have been executed.

Read More Answers.

Question # 4

A good software tester should Tolerance Chaos?

Answer:-

Tolerance for Chaos.
People react to chaos and uncertainty in different ways. Some cave in and give up while others try to create order out of chaos. If the tester waits for all issues to be fully resolved before starting test design or testing, she won't get started until after the software has been shipped. Testers have to be flexible and be able to drop things when blocked and move on to another thing that's not blocked. Testers always have many (unfinished) irons in the fire. In this respect, good testers differ from programmers. A compulsive need to achieve closure is not a bad attribute in a programmer-certainly serves them well in debugging-in testing, it means nothing gets finished. The testers' world is inherently more chaotic than the programmers'.
A good indicator of the kind of skill we arelooking for here is the ability to do crossword puzzles in ink. This skill, research has shown, also correlates well with programmer and tester aptitude. This skill is very similar to the kind of unresolved chaos with which the tester must daily deal. Here's the theory behind the notion. If you do a crossword puzzle in ink, you can't put down a word, or even part of a word, until you have confirmed it by a compatible cross-word. So you keep a dozen tentative entries unmarked and when by some process or another, you realize that there is a compatible cross-word, you enter them both. You keep score by how many corrections you have to make-not by merely finishing the puzzle, because that's a given. I've done many informal polls of this aptitude at my seminars and found a much higher percentage of crossword-puzzles-in-ink afficionados than you'd get in a normal population.

Read More Answers.

Question # 5

A good software tester should be Tenacity?

Answer:-

Tenacity.
An ability to reach compromises and consensus can be at the expense of tenacity. That's the other side of the people skills. Being socially smart and diplomatic doesn't mean being indecisive or a limp rag that anyone can walk all over. The best testers are both-socially adept and tenacious where it matters. The best testers are so skillful at it that the programmer never realizes that they've been had. Tenacious-my picture is that of an angry pitbull fastened on a burglar's rear-end. Good testers don You can't intimidate them-even by pulling rank. They'll need high-level backing, of course, if they're to get you the quality your product and market demands.

Read More Answers.

Question # 6

What makes a good software tester?

Answer:-

1. Know Programming. Might as well start out with the most controversial one. There's a popular myth that testing can be staffed with people who have little or no programming knowledge. It doesn't work, even though it is an unfortunately common approach. There are two main reasons why it doesn't work.
(1) They're testing software. Without knowing programming, they can't have any real insights into the kinds of bugs that come into software and the likeliest place to find them. There's never enough time to test "completely", so all software testing is a compromise between available resources and thoroughness. The tester must optimize scarce resources and that means focusing on where the bugs are likely to be. If you don't know programming, you're unlikely to have useful intuition about where to look.
(2) All but the simplest (and therefore, ineffectual) testing methods are tool- and technology-intensive. The tools, both as testing products and as mental disciplines, all presume programming knowledge. Without programmer training, most test techniques (and the tools based on those techniques) are unavailable. The tester who doesn't know programming will always be restricted to the use of ad-hoc techniques and the most simplistic tools.
Taking entry-level programmers and putting them into a test organization is not a good idea because:
(1) Loser Image.
Few universities offer undergraduate training in testing beyond "Be sure to test thoroughly." Entry-level people expect to get a job as a programmer and if they're offered a job in a test group, they'll often look upon it as a failure on their part: they believe that they didn't have what it takes to be a programmer in that organization. This unfortunate perception exists even in organizations that values testers highly.
(2) Credibility With Programmers.
Independent testers often have to deal with programmers far more senior than themselves. Unless they've been through a coop program as an undergraduate, all their programming experience is with academic toys: the novice often has no real idea of what programming in a professional, cooperative, programming environment is all about. As such, they have no credibility with their programming counterpart who can sluff off their concerns with "Look, kid. You just don't understand how programming is done here, or anywhere else, for that matter." It is setting up the novice tester for failure.
(3) Just Plain Know-How.
The programmer's right. The kid doesn't know how programming is really done. If the novice is a "real" programmer (as contrasted to a "mere tester") then the senior programmer will often take the time to mentor the junior and set her straight: but for a non-productive "leech" from the test group? Never! It's easiest for the novice tester to learn all that nitty-gritty stuff (such as doing a build, configuration control, procedures, process, etc.) while working as a programmer than to have to learn it, without actually doing it, as an entry-level tester.

Read More Answers.

Question # 7

A good software tester should be Honest?

Answer:-

Testers are fundamentally honest and incorruptible. They'll compromise if they have to, but they'll righteously agonize over it. This fundamental honesty extends to a brutally realistic understanding of their own limitations as a human being. They accept the idea that they are no better and no worse, and therefore no less error-prone than their programming counterparts. So they apply the same kind of self-assessment procedures that good programmers will. They'll do test inspections just like programmers do code inspections. The greatest possible crime in a tester's eye is to fake test results.

Read More Answers.

Question # 8

How to Identifying Software Quality Assurance Personnel?

Answer:-

Requirement Specification
Functional Specification
Technical Specification
Standards document and user manuals â€" If applicable (e.g. Coding standards document)
Test Environment Setup

Read More Answers.

Question # 9

A good software tester should Know the Application?

Answer:-

Know the Application.
That's the other side of the knowledge coin. The ideal tester has deep insights into how the users will exploit the program's features and the kinds of cockpit errors that users are likely to make. In some cases, it is virtually impossible, or at least impractical, for a tester to know both the application and programming. For example, to test an income tax package properly, you must know tax laws and accounting practices. Testing a blood analyzer requires knowledge of blood chemistry; testing an aircraft's flight control system requires control theory and systems engineering, and being a pilot doesn't hurt; testing a geological application demands geology. If the application has a depth of knowledge in it, then it is easier to train the application specialist into programming than to train the programmer into the application. Here again, paralleling the programmer's qualification, I'd like to see a university degree in the relevant discipline followed by a few years of working practice before coming into the test group.

Read More Answers.

Question # 10

A good software tester should be Hyper-Sensitivity to Little Things?

Answer:-

Hyper-Sensitivity to Little Things.
Good testers notice little things that others (including programmers) miss or ignore. Testers see symptoms, not bugs. We know that a given bug can have many different symptoms, ranging from innocuous to catastrophic. We know that the symptoms of a bug are arbitrarily related in severity to the cause. Consequently, there is no such thing as a minor symptom-because a symptom isn't a bug. It is only after the symptom is fully explained (i.e., fully debugged) that you have the right to say if the bug that caused that symptom is minor or major. Therefore, anything at all out of the ordinary is worth pursuing. The screen flickered this time, but not last time-a bug. The keyboard is a little sticky-another bug. The account balance is off by 0.01 cents-great bug. Good testers notice such little things and use them as an entree to finding a closely-related set of inputs that will cause a catastrophic failure and therefore get the programmers' attention. Luckily, this attribute can be learned through training.

Read More Answers.

Question # 11

A good software tester should be People Skills?

Answer:-

People Skills.
Here's another area in which testers and programmers can differ. You can be an effective programmer even if you are hostile and anti-social; that won't work for a tester. Testers can take a lot of abuse from outraged programmers. A sense of humor and a thick skin will help the tester survive. Testers may have to be diplomatic when confronting a senior programmer with a fundamental goof. Diplomacy, tact, a ready smile-all work to the independent tester's advantage. This may explain one of the (good) reasons that there are so many women in testing. Women are generally acknowledged to have more highly developed people skills than comparable men-whether it is something innate on the X chromosome as some people contend or whether it is that without superior people skills women are unlikely to make it through engineering school and into an engineering career. The fact is there and those sharply-honed people skills are important.

Read More Answers.

Question # 12

A good software tester should be Organized person?

Answer:-

Organized.
Can't imagine a scatter-brained tester. There's just too much to keep track of to trust to memory. Good testers use files, data bases, and all the other accouterments of an organized mind. They make up checklists to keep themselves on track. They recognize that they too can make mistakes, so they double-check their findings. They have the facts and figures to support their position. When they claim that there's a bug-believe it, because if the developers don't, the tester will flood them with well-organized, overwhelming, evidence.
A consequence of a well-organized mind is a facility for good written and oral communications. As a writer and editor, I've learned that the inability to express oneself clearly in writing is often symptomatic of a disorganized mind. I don't mean that we expect everyone to write deathless prose like a Hemingway or Melville. Good technical writing is well-organized, clear, and straightforward: and it doesn't depend on a 500,000 word vocabulary. True, there are some unfortunate individuals who express themselves superbly in writing but fall apart in an oral presentation- but they are typically a pathological exception. Usually, a well-organized mind results in clear (even if not inspired) writing and clear writing can usually be transformed through training into good oral presentation skills.

Read More Answers.

Question # 13

A good software tester should be Self-Sufficient and Tough?

Answer:-

If they need love, they don't expect to get it on the job. They can't be looking for the interaction between them and programmers as a source of ego-gratification and/or nurturing. Their ego is gratified by finding bugs, with few misgivings about the pain (in the programmers) that such finding might engender. In this respect, they must practice very tough love.

Read More Answers.

Question # 14

A good software tester should be Technology Hungry?

Answer:-

They hate dull, repetitive, work-they'll do it for a while if they have to, but not for long. The silliest thing for a human to do, in their mind, is to pound on a keyboard when they're surrounded by computers. They have a clear notion of how error-prone manual testing is, and in order to improve the quality of their own work, they'll find ways to eliminate all such error-prone procedures. Excellent testers re-invent the capture/playback tool many times. Dozens of home-brew test data generators. Excellent test design automation done with nothing more than a word processor, or earlier, with a copy machine and lots of bottles of white-out.

Read More Answers.

Question # 15

Professional Characteristics of a good SQA Engineer

Answer:-

Understanding of business approach and goals of the organization
Understanding of entire software development process
Strong desire for quality
Establish and enforce SQA methodologies, processes and Testing Strategies
Judgment skills to assess high-risk areas of application
Communication with Analysis and Development team
Report defects with full evidence
Take preventive actions
Take actions for Continuous improvement
Reports to higher management
Say No when Quality is insufficient
Work Management
Meet deadlines
Open Minded

Observant
Perceptive
Tenacious
Decisive
Diplomatic
Keen for further training/trends in QA

Read More Answers.

Question # 16

Personal Requirements For Software Quality Assurance Engineers

Answer:-

Challenges
Rapidly changing requirements
Foresee defects that are likely to happen in production
Monitor and Improve the software development processes
Ensure that standards and procedures are being followed
Customer Satisfaction and confidence
Compete the Market

Read More Answers.

Testing Most Popular Interview Topics.

1 : Manual Testing Frequently Asked Interview Questions and Answers Guide.

2 : QTP Frequently Asked Interview Questions and Answers Guide.

3 : JUnit Frequently Asked Interview Questions and Answers Guide.

4 : Software QA Frequently Asked Interview Questions and Answers Guide.

5 : QA Testing Frequently Asked Interview Questions and Answers Guide.

6 : Mobile Testing Frequently Asked Interview Questions and Answers Guide.

7 : Database Testing Frequently Asked Interview Questions and Answers Guide.

8 : Test Cases Frequently Asked Interview Questions and Answers Guide.

9 : Localization Testing Frequently Asked Interview Questions and Answers Guide.

10 : WinRunner Frequently Asked Interview Questions and Answers Guide.

About Global Guideline.

Global Guideline is a platform to develop your own skills with thousands of job interview questions and web tutorials for fresher's and experienced candidates. These interview questions and web tutorials will help you strengthen your technical skills, prepare for the interviews and quickly revise the concepts. Global Guideline invite you to unlock your potentials with thousands of **Interview Questions with Answers** and much more. Learn the most common technologies at Global Guideline. We will help you to explore the resources of the World Wide Web and develop your own skills from the basics to the advanced. Here you will learn anything quite easily and you will really enjoy while learning. Global Guideline will help you to become a professional and Expert, well prepared for the future.

* This PDF was generated from https://GlobalGuideline.com at **November 29th, 2023**

* If any answer or question is incorrect or inappropriate or you have correct answer or you found any problem in this document then don't hesitate feel free and e-mail us we will fix it.

You can follow us on FaceBook for latest Jobs, Updates and other interviews material.
www.facebook.com/InterviewQuestionsAnswers

Follow us on Twitter for latest Jobs and interview preparation guides
https://twitter.com/InterviewGuide

Best Of Luck.

Global Guideline Team
https://GlobalGuideline.com
Info@globalguideline.com