

The following applies for fresh hires

(Candidates with 0-6 months of experience)

All candidates looking for a role in programming are asked to write small programs on the white-board. The language/technology you are familiar with does **not** matter. Basic C programming skills are more than enough.

Being very strong in theory, but lacking in coding skills is a bad sign. Say for example, if the candidate is thorough in theory about various database normal forms - 1nf, 2nf etc. but is not able to write simple SQL queries, it does **not** impress us. The position of 'programmer' is strictly for people who can write good code.

You'll have to be very thorough in:

Basic Programming If asked to write programs for trivial problems, like say, sum of all even numbers between 200 and 400, you should be able to write a working program within 5-10 minutes - on the whiteboard/PC. You might also be given harder problems - which you can solve within 30-40 minutes.

Algorithms: Sorting, Trees etc. A Computer Science graduate is expected to be able to write simple sorts (Bubble/Insertion) fluently. Understanding advanced sorting methods and their use cases is a definite plus. Algorithms are a widely used topic during programmer interviews. Spending some time with the Introduction to Algorithms (Cormen et al) textbook would help a lot.

Databases: The candidates are asked build SQL queries for database schemas that are presented them. There will be questions from Networking, Operating Systems, Web Technologies and other computing areas. However the lack of knowledge in certain domains can be easily made up if you can write code and impress the interviewers with your capability to quickly learn, adapt and solve new problems.

A part of the interview will test your ability to learn stuff you already don't know. ie, given a hard problem that you don't have any clue, we watch how you set about solving it. You might be asked to explain the process through which you'll try to solve it. For such problems, you need not ultimately solve it - but what will interest us is 'how' you try to solve it.

Most of the simple programming questions are 'weeder' questions - which we use to quickly identify absolute non-programmers. They are very trivial questions - the kinds which are solved in any Computer Science graduate course (or even during High School). Here is a set of such sample questions:

- Reverse a sentence (“bob likes dogs” → “dogs likes bob”)
- List the first 100 prime numbers.
- Return distinct values from a list including duplicates (i.e. “1 3 5 3 7 3 1” → “1 3 5 7”)
- Implement a Linked List, insert values into it, and iterate through it.
- Write a program that shows the largest Fibonacci number that is less than an input value X.

If the candidate is asked to solve and mail us the solution for a programming problem before/after the interview, the code will be judged based on:

- Does it work?
- Is the code readable?
- Can a 3rd party read, understand and modify the code easily?
- Are the variables named properly?
- Does the code as a whole follow proper coding practices?
- Is the solution a right fit for the problem?
- Does the candidate complicate simple problems?