What to Expect in Your First Interview for a Programming Job

Written by a BCIT graduate, May 2004

Objective: *My goal here is strictly to give some examples of the type of questions which programmers can face in an interview situation. From talking to a lot of new grads, I realize a lot of people have not the slightest idea of the type of questions they will face. I wish to present a small subset of questions that may be asked. I do not wish to provide "canned" answers.*

Disclaimer: *This was written when I was a student here at BCIT and was doing some contract work for an IT company. I am not qualified to say whether an answer is correct or not (most of them are open-ended anyhow). Furthermore, I do not claim I know all types of questions nor the answers. However, I have been to quite a few interviews. My friends and I have collected a pool of interview questions from about 40 different companies. I have also read quite a few books on the programming interview process. Once again, the emphasis should be placed on the type of questions rather than the actual questions.*

1) The HR interview

The HR interview is usually the first interview, and is where the interviewer will have the chance to check your background and what it is that you would like to do. The interviewer will likely run down your resume chronologically and have you talk about each experience listed and ask you some questions on your corresponding interests. The interviewer may also ask you to show him/her your transcript. Examples of this type of question include:

- What are your interests?
- Why do you want to work for us?
- Are you creative?
- Are you intelligent?
- Which class did you enjoy the most?
- Why did you get such a bad mark in COMP XXXX?

2) The Technical Interview

Know your target. The type of questions asked is largely based on the company and the position for which you are applying. Usually but not always, a bigger company will be more inclined to ask you logic/puzzle questions along with other types of questions.

a) Knowledge-based Questions

Most companies will ask this type of question. The goal here is to simply verify that you know the stuff you say you know on your resume. Every skill listed on your resume is fair game and the interviewer can probe very deep in your understanding. For example, I have C/C++ and OOAD on my resume (as with almost every other programmer), some of the questions that I have been asked about C/C++ include:

- What is polymorphism?
- What is a real world example of polymorphism? Not the shape example.
- What is inheritance?
- What is the difference between public, private, protected, and virtual inheritance?
- How do you implement abstraction using C?
- When is multiple-inheritance appropriate?
- When do you use virtual methods?

Essentially, the interviewer can tell whether or not you know your stuff by simply listening to how much you can talk about it. Even if the interviewer doesn't know the specific technology listed on your resume,

he/she can simply ask you the what/why/how/when/where set of questions and then listen to you. As an example, I don't know CORBA well, but I can still answer questions about it.

- What exactly is CORBA? A standard? A technology?
- Why would you use CORBA over sockets (or other technology)?
- How do you implement CORBA?
- When is it better to use CORBA than some other competing technology?
- Where is CORBA being used (what industry)?

Because of the ease of asking deep questions, you should NEVER LIE on your resume. However, a big decision that you have to make is whether to list a skill that you may have but in which you are not an expert. For example, I have done Java programming in the past for a couple of courses, but if I'm probed deeply by the interviewer about it, it may hurt me more than help me (EJB, J2EE, etc…). According to *resource i*, it is better to close down on the scope of the technology. So what I should do is list Java Net classes, J2SE jdk 1.2 and list Java under the programming language section. This will gear the questions more towards what I know rather than the broad Java technology. An interview can easily go an hour or more just based on questions from your technology skills on your resume.

b) Logic/ Puzzle Questions

Puzzle questions were mostly popularized by Microsoft. There is a book dedicated to questions based on this *(resource ii)*. The idea is to see how creative you are and whether you can reason logically. A good number of these are very open-ended. These types of questions are roughly grouped into the following:

i. Statistics

These types of questions test to see if you can apply basic statistics to problems. Examples of these include: There are two bullets in six chambers of the two. I pull the trigger, it's empty one. Now before I pull the trigger again at your head, would you like me to spin or not?

- Mike and Todd have $21 between them. Mike has $20 more than Todd. How much does each have? You can't use fractions in the answer.
- Give me a string at random from this file (out of N strings). What if you can only make one pass?

ii. Fermi's Estimation

Estimation is actually quite important for a programmer, whether you're estimating budgets, resources, and project deadlines. That's what the following examples are testing.

- How many gas stations are there in the US/Canada?
- How many piano tuners are there in the world?

iii. Puzzle / Design Questions

- How would you convince Google to switch from Linux to Windows? (Microsoft question)
- Which way should the key turn to unlock the car door?

c) Programming/ Algorithm Questions

Depending on the company, the bulk of your interview questions may be spent here. The idea is that as long as you know the fundamentals, they can teach you the technology (knowledge-based). I recently had an interview where the first question they asked me was right below. The suggestion is to know the fundamentals in each of these and not to madly memorize the answers. Know your data structures well.

String Manipulation

- Reverse the characters in a string.
- Reverse the words in string. For example, "hello world" becomes "world hello"
- Delete one instance of string from another.

Linked list manipulation

- Given a singly linked list, write a deletion function for it.
- Given a singly linked list, return the mth from last member.

Big O

- What is the Time complexity of printing all the values in a Binary Tree?

Bit manipulation

- Write a one-liner to determine if x is a power of two.
- Write a one-liner to determine if x is an even number.
- How do you count the number of 1-bits in a number?

Logic tests

You can find samples of these on the Internet, just Google for IQ tests. I have taken one logic test where it's flow-chart based, but any programmer shouldn't have any problem with these.

Other types

They could ask you to debug a program right on the spot. Or they could ask you to optimize the code. Some other popular types of questions involve recursion, or trees. Here's a funny one that I've heard:

- What is your favourite tree in the world?
- What's your favourite programming language? (be careful)

d) Software Engineering Questions

Answer these questions based on your experience, ideally from projects that you have done. It's always good to give an answer of a good software engineering practice and then give an example from your previous projects/ experiences.

- Discuss the difference between RUP vs. XP.
- Rank in order of your preference: Software development, technical support, and software testing.
- Why do so many projects fail, and have you had any failed projects?

3) The Programming Assignment

In the interview I had for my last job, I was asked to write a simple kernel module right on the spot. I was lucky enough to have written device drivers for fun a week before, so it was still fresh in my memory. The key to these programming assignments is to remember to document, comment well, and include any design decisions/assumptions/limitations of your program. Some programming assignments include:

- Write a program in any language that prints out the cards in a deck, shuffle the deck of cards, and print it out again.
- Write a program that outputs its own source code.

4) Other issues/advice

These are collected from various resources and paraphrased:

- Even if you don't have all the skills required in the posting, you should give it a try anyways. Most of the companies list what they'd like to have rather than what is actually required. I once read a posting for 5 years of C++ experience back in 1999. At the same time, if your skill set really doesn't match the position's skill set, you probably shouldn't waste your or their time.

- Don't send in your resume bloated with buzzwords in which you don't actually have experience. It destroys your credibility as soon as the interviewer finds out you don't know what you say you know and the rest of the interview can go sour.

- Don't just memorize answers, it may be counterproductive. Think about the sheer amount of different variety of string manipulation questions alone that they can ask. It is therefore more productive to know the fundamentals. Furthermore, if you memorized a whole bunch of programming questions and you get one that you've never heard of before, you may start to panic (happened to my friend). The general advice is to practice the questions, but don't memorize the code.

- Don't try to bluff your way through a knowledge-based question to which you don't know the answer. If you have bloated your resume and they ask a specific question about a specific skill that you don't actually have, you may be better off saying that you haven't used the technology for a couple of years and that the resume hasn't been updated accordingly or that you need a refresher course.

- A lot of questions are very tough, and the interviewer may not actually expect you to get the answers correct. Keep on asking questions and keep on thinking out loud. Most interviewers are nice and want you to succeed. Why would they waste their time on you if they didn't?

5) The Resume

The rule of thumb is to keep the resume one page if your industry experience is under 5 years. Please see your Career or Employment Centre for help and advice.

6) Resources

i. Ace the Technical Interview: Michael Rothstein, 2000, ISBN: 0072126221
   - This book is great for knowledge-based questions. However I find that the questions do not go as deep as some real interview questions.

ii. How Do You Move Mount Fuji?: William Poundstone, 2003, ISBN: 0316919160
   - This is a must-read if you're going to interview at Microsoft, or any of the big companies (Amazon.com, IBM, etc…).

iii. http://halcyon.usc.edu/~kiran/msqs.html
   - This site contains a lot of programming interview questions, more than you can probably do. Remember the idea is know the concepts/basics, and not to memorize the answers.

iv. The Practice of Programming: Brian Kernighan, 1999, ISBN: 020161586X
   - This book contains general programming knowledge from the godfather of C. It also has many exercises that turn out to be good interview programming questions.

7) Conclusion

I hope I have contributed in some way to your understanding of the programming interview. I sincerely hope that people here will help each other out and pass on leads and information. If a posting really doesn't match your skill set, why not pass it on to a friend who is more compatible? We are all in this profession because of our love for technology/coding and once we find our dream job, it will all be worth it. It took me more than 5 months to find a job the first time around (3 years ago and I'm starting to look for one soon), and it took some of my friends even longer. The most important things are to keep optimistic and keep persevering. Have an awesome attitude, use the resources available to you, and of course, keep on coding for the fun and love of it.