How To Ace A Paired Programming Interview

"Paired programming" is something of a catch-all term. Any two (or three, *maybe* four) people working together to write some code qualifies as pair programming.

There are many different approaches to pairing. You might adopt the "driver/navigator" strategy in which one person does the typing and the other person dictates the direction of the code. You might try "ping-pong" programming in which one person types for a few minutes, then you switch and let the other partner take over.

While there are many approaches to pairing, the goals are usually the same:

- Two brains are better than one. When you have two people strategizing together, they'll come up with a more robust and elegant solution than one person alone.
- Spread the knowledge around. When two people are responsible for building a specific feature, it's easier to spread the knowledge of how that feature works to the rest of the team. This makes it easier to grow the feature, refactor it or rebuild it in the future.

Why Is Pairing Part of the Interview Process?

There are two reasons why pair programming has become a popular interview tool:

1. Pairing is a common and useful tool that you'll likely use in your new job as a developer. Many companies want to see how you perform when pairing.
2. Pairing is a great way to learn how a candidate thinks and what they're like to work with.

The pairing challenge can be an intimidating part of the interview process, but it doesn't have to be. Programming is all about solving problems. Your potential future teammates are trying to find out what it's like to solve a problem with you, because that's what you'll be doing in your job every day.

The most important thing to keep in mind is that this section of the interview is designed for your potential teammates to learn what it's like to work with you. They want to know how you break down and come to understand the problem or the task at hand, how you think through possible solutions and what it's like to collaborate with you throughout that process. They want to collaborate, they want to ask and answer questions and they want to have fun.

What Does Pair Programming Look Like in an Interview?

The manner in which you will be asked to pair with other developers during your interview can vary. Here are a few common options:

- Set up a development environment and build out a sample feature like at Cognizance.
- Work with an employee to build out a sample feature on the company's code base like at Vulcan Labs.
- Pick a task, code and talk through your understanding of the problem to be solved with your partner like at Silex Consulting.
- Solve a code challenge together like something you might see on the Euler Project.
- Hunt down and fix the bug presented to you in a given code base.

Although the specific challenge you are asked to pair can be different, the overall approach to crushing this part of the interview stays the same.

By adding the strategies outlined below to your toolkit, you can successfully illustrate your problem-solving skills, effectively collaborate with your pairing partner, and ace this part of the interview.

How To Pair in Your Interview

There are a few strategies you can use to ace the pair programming interview:

1. Be prepared
2. Be yourself
3. State what you know
4. Ask questions
5. Start simple
6. Write tests

1. Be Prepared

If you're not sure what to prepare, ask!

If pairing remotely, you will be using software that allows you to talk to each other and share your screen eg. Zoom or Google Hangouts. There are also plugins for Atom, Sublime, VS Code and almost any other code editor that allow you to share your IDE with the other developer and modify the code at the same time. You can also allow the other developer remote access to your computer.

2. **Be Yourself**

This may sound like generic advice, but it is much more important for a pair programming session compared to a general or technical interview. Why? Simply because, in some general interviews, an HR person talks to you and gauges your personality for the duration. While you would be in the same company as they are, you won't be directly working with them every day.

In a pair programming session, if the company does pair programming most of the time anyway, you would likely be working closely with your interviewer as part of your job. That is the main difference. Like in any relationship, it's hard to have a long term relationship if you build it based on only part of the picture of who you are.

3. **State What You Know**

When the challenge is first presented to you, it can feel overwhelming. Start by repeating back to the interviewer what you understand the challenge to be. This will help you to clarify your understanding of what you're being asked to do. It will also give the interviewer the opportunity to offer you more information and start collaborating with you right off the bat.

Remember that a big reason *why* you are pairing during your interview is so that your future teammates can find out what it's like to work with you. So give them lots of opportunities to do just that! Stating what you know right away will provide your partner with a chance to immediately provide feedback and help you better understand the task at hand.

Once you've stated your initial understanding of the challenge and begun working together with your partner, it probably won't be 100% smooth sailing. You'll start to work on a solution, realize another problem or a greater degree of complexity, and have to expand your solution from there.

These moments are a great time to fall back on the "state what you know" strategy. Repeat out-loud what your solution accomplishes so far, and repeat out-loud what challenges you still need to solve. This does three things:

- Demonstrates your understanding of your code and the challenge to your partner
- Allows you to take a step back and see problem more clearly, making the solution more readily apparent to you
- Provides your partner an opportunity to jump in and offer their own ideas and solutions

4. **Ask Questions**

Stating what you know goes hand-in-hand with another strategy: asking questions. You can follow up statements about what you know with a question about what you don't. Pair programming is all about collaboration. Your future teammate wants to know what it is like to work with you and learn more about how you solve problems.

Pair programming is *NOT* about quizzing you, putting you on the spot to prove your knowledge or catching you off guard. So, ask lots of questions! Ask your partner to give you examples of how a user might interact with the feature you're building; ask them to clarify what they mean by a particular statement; ask them if they have any ideas or preferences of where to get started.

Asking questions does a few things for you:

- Allows your partner to understand how you approach a given problem
- Allows your partner to collaborate with you by providing answers, ideas or follow up questions
- Gives you the additional information you need to solve the problem

Pro-Tip:

It's okay to take a moment to think. The thought of having 'dead air' during an interview is really scary. You can let your partner know that you will need a bit of time to think and that you will let them know your solution after. This shows that you are acknowledging their presence and that you will be communicating your thoughts after you process them. Communication is key!

5. **Start Simple**

When faced with a complex challenge, like solving a logical puzzle or building a new feature, it can be easy to jump to the most complicated scenario first. Resist the urge to start building a solution that will address *all* the requirements or potential scenarios of the challenge before you. Instead, break the problem down into the smallest possible parts and start with the most simple scenario. This will give you somewhere to start, allow you to get to work with your partner and give you a deeper understanding of the problem at hand. This understanding will allow you to expand your initial solution and solve the problem.

6. **Write Tests!**

Not every pair programming challenge is a good fit for this piece of advice. However, whenever possible, write tests for your code! This is an important strategy for a few reasons:

- Testing allows you to think through the different possible scenarios that your solution should address
- It allows you to test your code (obviously). Testing is often overlooked in interview pairing but it's an easy way to make sure that your code behaves like you expect it to
- It shows your commitment to coding best practices.

Resources:

- https://dev.to/sophiedebenedetto/pair-programming-for-interviewees-1eem
- https://www.freecodecamp.org/news/things-ive-learned-from-pair-programming-interviews-35a4db7d7443/