COMPETITIVE CODING BASICS

Firstly, you have to get your implementation right

It’s time to solve some problems. Start with simple ones. You have already been implementing programs for two weeks. You would have seen that it is not always easy to implement your logic in code. This stage is primarily to define your coding style and get comfortable with automating logics. Maybe you like to code with spaces, maybe with tabs. You may like to put your braces on the same line of the function definition or you may want to put it on the next line. Maybe you are more comfortable with Java, maybe with Python.

Whatever the case, a little practice goes a long way! Make sure to practice solving problems on a regular basis.
And keep in mind these two principles while developing your coding style.

Easy to implement. You should feel comfortable implementing the solution you came up with. Why? Because during the competition, the last thing you want to happen is to get lost in your code. It’s always better to think 15 more minutes about implementation rather than spending 10 more minutes coding it.

Easy to read. ie, ‘Easy to debug’. Let’s face it, we both know that bugs appear all the time. Do you know that feeling when you have 10 minutes left and you don’t find that bug? Yes, you do. To solve that you have to write legible code. So when you start debugging, the code would feel natural and simple to follow.

How to boost your implementation skills?

Practice, practice and more practice. Top 250 solved problems on SPOJ are recommended for this stage. Solve them in that exact order. And remember to think of the solution before you actually get down to implement it.
Before you say ‘this is too hard for me, I will try the next one’, it is advisable to use a paper and a pen to *think out* the solution. Even if you really cannot solve it, you will develop algorithmic thinking for sure. And honestly speaking, that is the real learning.
If you cannot get ahead after a while (minimum thirty minutes though), look in the forums and editorials for help.

Next, master Data structures and try different algorithms.

If you work easy problems, you will never become better.

The most effective way to find what you don’t know is to actually encounter it. As a rule of thumb, every 3 problems you solve, one should teach you something new. If not, choose them more carefully. Choose harder problems!

After you finish those 250 problems from SPOJ, you will have an overview of the main topics of competitive programming. By deeply understanding the logic behind basic algorithms, high-level algorithms will seem easy to understand. So you can rapidly leverage your knowledge.

Coding Tips

- Read the problem carefully. Usually, the problem is verbose but the algorithm is small.

- Work out your algorithm with the sample input and output. Sometimes, an explanation might also be given to assist you.

- *Always* start on paper. Work out the problem in your mind, try a few small inputs and outputs and build the logic.

- Make sure before you start coding you have a grasp of the logic and the sample output matches the output of your logic

- Pick a language and stick with it. This ensures you become better in that language as well.

- Trial and error is your best friend. Print out every result to see what your code is doing.

- Time is limited in competitive programming and more efficient algorithms are preferred.

- If something can be solved by math, use math.

- Lastly, never lose hope! Google is your second best friend.