How to become a better ICPC team?

(Except for improving problem solving skills)

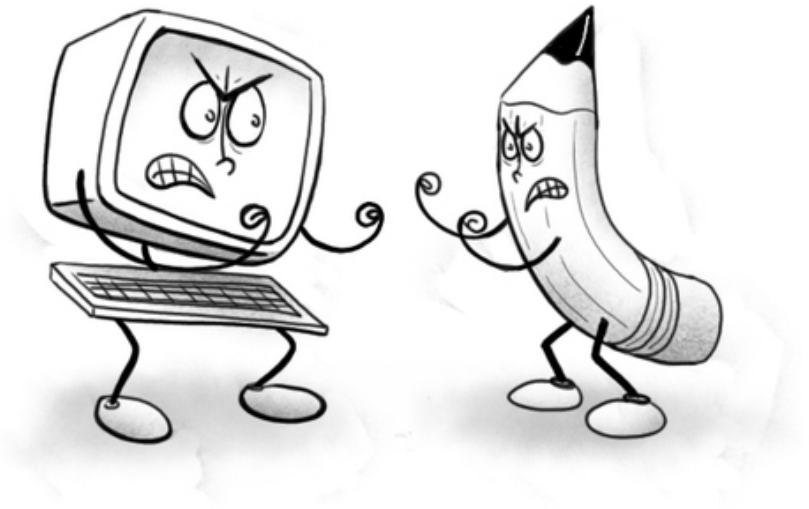
How to become a better team?

- Improve your team work
- Computer bottleneck:
  - Write code on paper
- Time bottleneck:
  - Help teammates with testing
  - Choose and assign your problems

Learn from your teammates & Improve your teamwork

1. Discuss your individual abilities.
   a) One by one, describe your experience in the competitions. Describe what you believe your strengths and weaknesses are.
   b) Try to suggest ideas to help teammates improve, and try to discuss how your teamwork can help exploit your strengths and cover for your weaknesses.
2. Discuss your teamwork.
   a) How can you work better as a team?

Computer bottleneck



- Don’t hog the computer!
  - If another teammate needs the computer and you are taking longer and longer, decide on a deadline and then let go.

- When your teammate is using the computer, write code on paper.
  - More detailed code on paper saves computer time.
    Prefer writing code that compiles over pseudo-code.

Testing

- When can testing help?
  - When designing the solution, to make sure it works
  - When your teammate is solving, and you have time to support
  - When you are ready to submit, and want to avoid a non-accepted submission
  - When you submit, and the solution is not accepted

- What to include?
  - Include the sample test cases
  - Include corner cases
    - Think like the problem author: try to come up with the worst inputs, try to find cases that are hidden in the **description**, think of **extreme** values (0, 1, negative, large).

Tips for testing

- Read the problem carefully again.
- Verify that you reset everything.
  If your program needs to support multiple test cases in a single run, include two identical sample test cases consecutively.
- Check for white spaces and minor mistakes in output.
  Let the computer compare for you; don't settle for looking at the solution.

Linux

```
./a.out < input > myoutput
diff myoutput expectedoutput
```

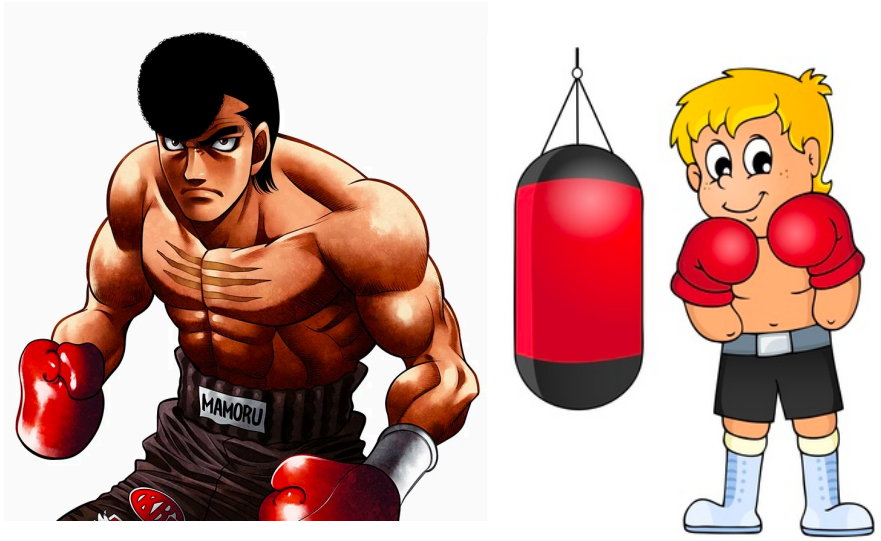
Windows

```
solution.exe < input.txt > myoutput.txt
fc myoutput.txt expectedoutput.txt
```

- Check for nasty input formatting.
  If the problem description doesn't impose nice formatting, try to add spaces and tabs to your test cases.
- Check for overflows, out of bound errors, or an inefficient solution.
  Include large test cases: cases with trivial structure that is easy to verify (verify the correctness of the solution), and random cases (verify ending in reasonable time, and that the output looks reasonable).

Choosing your battles

- If you didn’t find an easy problem and didn’t read all of them, **keep reading!** You don’t want to waste time on harder problems.



- If you find a problem that you think will be easier for another teammate, propose them to work on that.
  - Try to summarize the problem to save them reading time.