Getting Started with Programming Challenges

Arnaud Malapert, Gilles Menez, Marie Pelleau

Master Informatique, Université Côte d'Azur

Course Goals

- To provide a challenging, self-motivating course for good students to learn what makes programming/computer science fun and exciting.
- To strengthen the algorithmic/procedural intuition of students raised in a world of object-oriented programming.
- To provide an enthusiastic cadre of good students, and perhaps strengthen student culture.
- To prepare for interviews and progress in your careers.

Should you be taking this course?

It depends upon your interests, background, and skills.

Class Participation!

This course will require class participation!

- The first part of each lecture will introduce the problems and relevant theory.
- In the second part, you will let you try to solve them.
- No effort and/or no discussion equals boring class!

About the ACM Contest

Association for Computing Machinery (ACM)

It brings together computing educators, researchers, and professionals to inspire dialogue, share resources, and address the field's challenges. As the world's largest computing society, ACM strengthens the profession's collective voice through strong leadership, promotion of the highest standards, and recognition of technical excellence.

ACM International Collegiate Programming Contest (ICPC)

It provides college students with opportunities to interact with students from other universities and to sharpen and demonstrate their problem-solving, programming, and teamwork skills. The contest provides a platform for ACM, industry, and academia to encourage and focus public attention on the next generation of computing professionals as they pursue excellence.

About the ACM Contest

The contest stresses teamwork (3 people with 1 or 2 computers per team) as well as individual efforts, since small programs are best written by one person, perhaps after group discussions.

- Many of the problems are well-known exercises couched in different guises.
- The judges provide very little feedback about why your program is wrong.
- Often you must debug it by reading the specifications again.

Contest Scoring

1. **The team score is the number of problems solved correctly** over the course of the contest, typically 4 hours.
2. Ties are broken by the cumulative elapsed time taken to correct submissions, with possible time penalties given for each incorrect submission of an (ultimately) correctly solved problem.

Problem Specification: DOUGHNUT

Problem description

Young Harry was asked to buy some foodstuff to his neighbour, weird old lady who owned a lot of fat cats. But cats were weird too and they ate only doughnuts. So the lady wanted . . .

Input Description

There is a single positive integer t ($t \leq 100$) on the first line of input which corresponds to the number of tests (Harry was asked to buy doughnuts few times). Then t lines . . .

Output description

t lines containing word "yes" if Harry is capable of handling the task or "no" if doughnuts would cause his back crack.

Input and Output Examples

Problem Solution

A solution is a program in whatever languages you wish

- without parallelism (no threads),
- without external library calls,
- without file/network IO.

Problem Solution

A solution is a program in whatever languages you wish

- without parallelism (no threads),
- without external library calls,
- without file/network IO.

The solution program must

- read the test case(s) from the standard input;
- solve the test case(s);
- print the results on the standard output.

Of course, these steps can be imbricated.
This is the famous a **Read-Eval-Print Loop**.

Problem Solution

A solution is a program in whatever languages you wish

- without parallelism (no threads),
- without external library calls,
- without file/network IO.

The solution program must

- read the test case(s) from the standard input;
- solve the test case(s);
- print the results on the standard output.

Of course, these steps can be imbricated.
This is the famous a **Read-Eval-Print Loop**.

Standard Streams

They are preconnected input and output communication channels between a computer program and its environment when it begins execution.

Feedback from the (Automatic) Judge I

Be aware that the judges are often very picky as to what denotes a correct solution. It is very important to interpret the problem specifications properly and not make assumptions.
The judge is likely to return one of the following verdicts:

Accepted (AC)

Congratulations!

Wrong Answer (WA)

Your program returned an incorrect answer to at least one secret test case.

Time Limit Exceeded (TLE)

Your program took too much time on at least one of the test cases, so you likely have a problem with efficiency.

Feedback from the (Automatic) Judge II

With a good coding methodology, the judge should rarely return:

Compilation Error (CE)

The compiler could not figure out how to compile your program. Warning messages are ignored by the judge.

Runtime Error (RE)

Your program failed during execution due to a segmentation fault, floating point exception, or similar problem.

Memory Limit Exceeded (MLE)

Your program tried to use more memory than the judge's default settings.

Output Limit Exceeded (OLE)

Your program tried to print too much output, perhaps trapped in a infinite loop.

Languages

Automatic judges

Robot judges accept programs in C, C++, Java, Python, and Gnu R. In fact, You may use whatever language you wish.

Popular languages

- C/C++ is still the most popular language for programming contest.
- Python often comes second.

Choose the language that you like the most as default.

Training for the challenges

Private Platform: for the labs

sphere-engine.com

Local Contest: for the graduation

Read the team's guide of the pc2 software.

Worlwide Contest

In 2018, the university has organized hubs for the Google Hashcode.

Public platforms: just for fun

- spoj.com.
- uva.
- codechef.com.
- and many others.