Lecture 1:
Getting Started

Steven Skiena

Department of Computer Science
State University of New York
Stony Brook, NY 11794–4400

http://www.cs.sunysb.edu/∼skiena

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 for the ACM Programming Team, and perhaps strengthen student culture.

Should you be taking this course? It depends upon your interests, background, skills, and anticipated graduation date.

Administrivia

Make sure I get your name and *email* address written clearly, as well as whether you intend to take this course for credit.

Be sure to register on *both* the Programming Challenges *and* Univ. de. Valladolid robot judges. All programs must eventually be submitted over http://www.programming-challenges.com, but this way you can work even if there is trouble with one judge.

Get the textbook as soon as possible, for we will be following it very closely.

Class Participation!

This course will require class participation!

The first lecture of each week will introduce the problems and relevant theory.

The second class period we will discuss *your* efforts to solve them.

No effort and/or no discussion equals boring class!

Think John Cage's masterpiece *4'33"*…

About the ACM Contest

The International Programming 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.

ACM ICPC Scoring

The team score is the number of problems solved correctly over the course of the contest, typically 5 hours.

Ties are broken by the cumulative elapsed time taken to correct submissions, with time penalties given for each incorrect submission of an (ultimately) correctly solved problem.

Stony Brook typically does very well in the Greater New York Region, reaching the finals in 2006 and 2009!

Feedback from the Judge

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!
- *Presentation Error (PE)* – Check for spaces, left/right justification, line feeds, etc.
- *Accepted (PE)* – Your program has a minor presentation error, but the judge is letting you off with a warning. Stop here and declare victory!

- *Wrong Answer (WA)* – Your program returned an incorrect answer to one or more secret test cases.
- *Compile Error (CE)* – The compiler could not figure out how to compile your program. The resulting compiler messages will be returned to you. 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. Check for invalid pointer references or division by zero.
- *Submission Error (SE)* – You did not correctly specify one or more of the information fields, perhaps giving an incorrect user ID or problem number.

- *Time Limit Exceeded (TL)* – Your program took too much time on at least one of the test cases, so you likely have a problem with efficiency.
- *Memory Limit Exceeded (ML)* – Your program tried to use more memory than the judge's default settings.
- *Output Limit Exceeded (OL)* – Your program tried to print too much output, perhaps trapped in a infinite loop.
- *Restricted Function (RF)* – Your source program tried to use an illegal system function such as `fork()` or `fopen()`. Behave yourself.

Languages

Both robot judges accept programs in C, C++, Pascal and Java. You may use whatever language you wish.

C++ is still the most popular language.

Be aware that many students have had difficulty using Java on the judges, largely due to version incompatibility, although this is improving.

Using standard IO in Java is somewhat difficult – a 35-line standard IO template can be downloaded from the judge.

Verdict by Language

It is interesting to tabulate the judge's verdicts by programming language:

Lang	Total	AC	PE	WA	CE	RE	TL	ML	OL	RF
C	451447	31.9%	6.7%	35.4%	8.6%	9.1%	6.2%	0.4%	1.1%	0.6%
C++	639565	28.9%	6.3%	36.8%	9.6%	9.0%	7.1%	0.6%	1.0%	0.7%
Java	16373	17.9%	3.6%	36.2%	29.8%	0.5%	8.5%	1.0%	0.5%	2.0%
Pascal	149408	27.8%	5.5%	41.8%	10.1%	6.2%	7.2%	0.4%	0.4%	0.5%
All	1256793	29.7%	6.3%	36.9%	9.6%	8.6%	6.8%	0.5%	1.0%	0.6%

110101 (The 3n+1 problem)

The Collatz or Hailstone number sequences, a famous open problem in number theory. Why doesn’t it ever cycle?

110102 (Minesweeper)

You are asked to compute adjacencies from mine positions, which is in principle straightforward. But what about computing mine locations from partial adjacency counts? It is NP-complete!

110103 (The Trip)

How do we minimize the flow of money in balancing expenses after a trip? Who gets the extra pennies if the total does not divide evenly?

110106 (Interpreter)

Build an interpreter program for a very simple computer architecture. How do we extract digits from the integers to make parsing easier?