Team Manual for the ETH+EPFL Local Contest 2013

by Robert R. Enderlein*

Here follows a short summary of the system interface. This is meant as a quick introduction, to be able to start using the system. It is however strongly advised that you read all of this manual. There are specific details of this jury system that might become important when you run into problems.

Domjudge runs through a web interface that can be found at `https://ec2.hc2.ch/team`. See Figures 1, 2, 3 for an impression.

Reading and writing

Solutions have to read all input from 'standard in' and write all output to 'standard out' (also known as console). You will never have to open (other) files. See Section 11 for some examples.

Submitting solutions

From your team "Overview" page, `https://ec2.hc2.ch/team`, click "Select file..." in the left column and select the file you want to submit. By default, the problem is selected from the base of the filename and the language from the extension. See Figure 1.

Problem statements and supporting material

The problem statements and supporting material are available on the "Problem statements" tab of your team page. We also provide a link to the allowed documentation on that page.

Participating from home

If you are not participating from the official ETH or EPFL contest locations then you must either:

- Use a VPN to get an IP address in 129.132.0.0/16 or 128.178.0.0/16.
- Or contact the organizers about half an hour before the start of the contest and tell them your public IP address (google for "my ip"). (You can also ask later, but then you might not be able to start the contest in time.)

Otherwise, the firewall of the contest server will not let you though.

*This document was adapted from the Domjudge team manual.

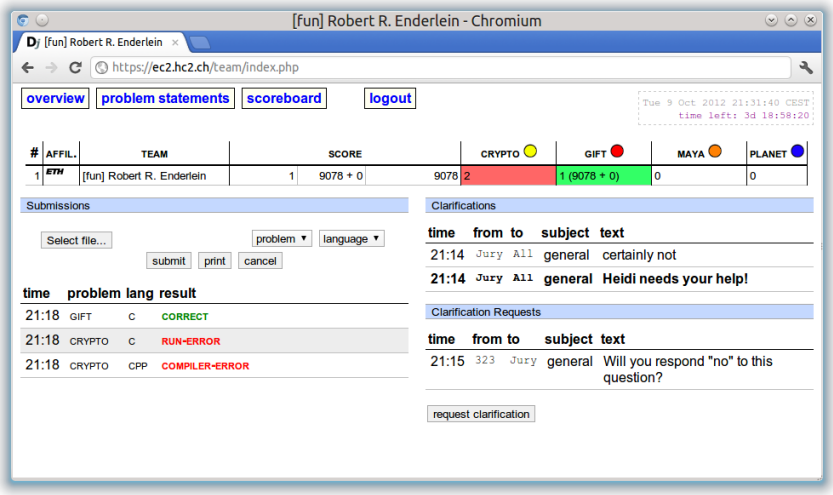

Figure 1: The team web interface overview page.

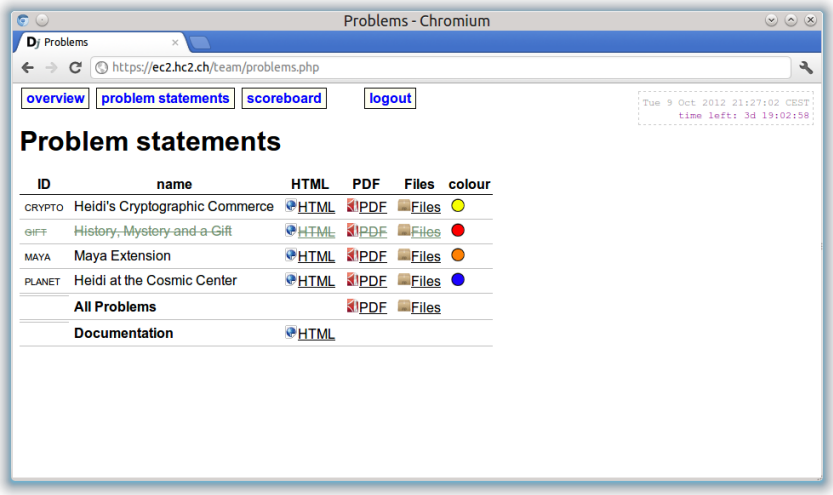

Figure 2: The problem statements page.

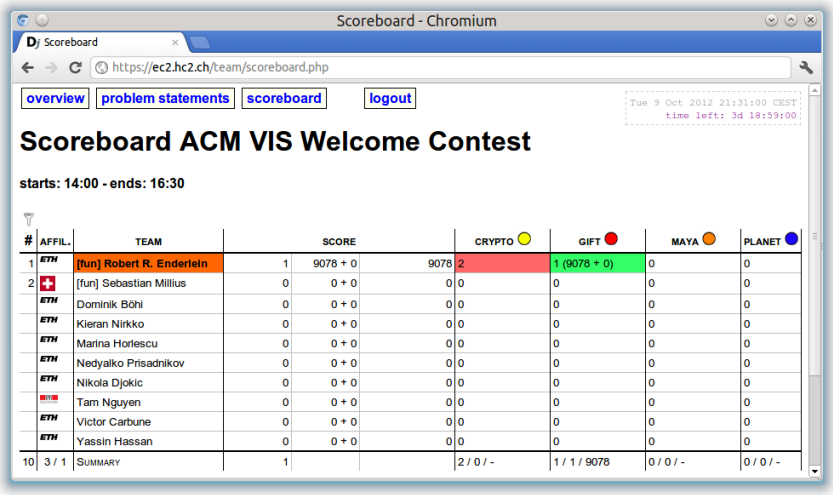

Figure 3: The scoreboard webpage.

1 Accessing the Server

The contest server's URL is `https://ec2.hc2.ch/team`.

The server is protected by a firewall. Your public IP must be in 129.132.0.0/16 or 128.178.0.0/16 in order to be able to access the server. If you are not participating from the official contest locations, then we kindly ask you to either:

- Use the VPN of ETH or EPFL.
- Or contact the organizers about half an hour before the start of the contest and tell them your public IP address (google for "my ip"). (You can also ask later, but then you might not be able to start the contest in time.)

2 Submitting solutions

Solutions must be submitted through Domjudge's web interface at `https://ec2.hc2.ch/team`. If this is the first time you go to that page you will need to login with your username and password (which was sent to you per e-mail when you registered). If you have trouble accessing the interface, please raise your hand and wait for an organizer to come and assist you.

In the left column click "Select file..." to select the file for submission. Domjudge will try to determine the problem and language from the base and extension of the filename respectively. Otherwise, select the appropriate values.

After you hit the "submit" button and confirm your submission, you will be redirected back to your submission list page. On this page, a message will be displayed that your submission was successful and the submission should be present in the list. An error message will be displayed if something went wrong.

3 Viewing the results of submissions

The left column of your team web page shows an overview of your submissions. It contains all the relevant information: submission time, programming language, problem and status. The address of your team page is `https://ec2.hc2.ch/team`.

The top of the page shows your team's row in the scoreboard: your position and which problems you attempted and solved. Via the menu you can view the public scoreboard page with the score from all the teams. One hour before the end of the contest (half an hour during the "Welcome contest") however, the scoreboard will be "frozen"; the full scoreboard view will not be updated anymore, but your team row will. Your team's rank will be displayed as "?".

3.1 Possible results

A submission can have the following results:

Verdict	Description
Correct	Your program passed all tests: you solved this problem!
Compiler-error	There was an error when compiling your program. On the submission details page you can inspect the exact error.
Timelimit	Your program took longer that the maximum allowed time for this problem. Therefore it has been aborted. This might indicate that your program hangs in a loop or that your solution is not efficient enough.

Run-error	There was an error during the execution of your program. This can have a lot of different causes like division by zero, incorrectly addressing memory (e.g., by indexing arrays out of bounds), trying to use more memory that the limits, etc. Also check that your program exists with exit code 0!
No-output	The judge did not produce any output. This probably means your program exited early.
Wrong-answer	The output of your program was incorrect. This can happen simply because your solution is not correct, but remember that your output must comply exactly with the specifications of the jury.
Presentation-error	The output of your program has differences in presentation with the correct results (for example in the amount of whitespace). This will, like WRONG-ANSWER, count as an incorrect submission.
Too-late	Bummer, you submitted after the contest ended! Your submission is stored, but will not be processed anymore.

4 Clarifications

All communication with the jury is to be done with clarifications. These can be found in the right column on your team page. Both clarification replies from the jury and requests sent by you are displayed there.

There is also a button to submit a new clarification request to the jury. This request is only readable for the jury and they will respond as soon as possible. Answers that are relevant for everyone will be sent to everyone.

It is your responsibility to check the clarifications column regularly.

5 How are submissions being judged?

5.1 Submitting solutions

With the web interface, you can submit a solution to a problem to the jury (see Section 2). Note that you have to submit the source code of your program (and not a compiled program or the output of your program). There your program enters a queue, awaiting compilation, execution and testing on one of the jury computers.

5.2 Compilation

Your program will be compiled on a jury computer running Ubuntu Linux (32 bits).

The compilation commands are as follows:

C	gcc -Wall -O2 -pipe -o $DEST $SOURCE -lm -lgmp
C++	g++ -Wall -O2 -pipe -o $DEST $SOURCE -lgmp -lgmpxx
Java	javac -d . $SOURCE

where $SOURCE represents the file you submitted.

5.3 Testing

After your program has compiled successfully, it will be executed and its answer compared to the answer of the jury. Before comparing the output, the exit status of your program is checked: if your program exits with a non-zero exit code, the verdict will be "Run-error". There are some restrictions during execution; if your program violates these, it will also be aborted with a "Run-error".

The command for executing Java programs is as follows:

Java	java -Xrs -Xss8m -Xmx98304k $MAINCLASS

where $MAINCLASS is the name of the class of the jury library.

5.4 Restrictions

To prevent abuse, keep the jury system stable and give everyone clear and equal environments, there are some restrictions to which all submissions are subjected:

Compile time	Compilation of your program may take no longer than 30 seconds. After that compilation will be aborted and the result will be a compile error. In practice this should never give rise to problems. Should this happen to a normal program, please inform the jury right away.
Source size	The source code of your program may not exceed 256 kilobytes, otherwise your submission will be rejected.
Memory	During execution of your program, there are 64 megabytes of memory available (96MB for Java). This is the total amount of memory (including program code, statically and dynamically defined variables, stack, ... — The Java VM will not be counted however). If your program tries to use more memory, it will abort, resulting in a run error. The stack size is limited to 8MB for all languages.
Program output	You are not allowed to output more than 4 megabytes on standard error, and will get a "Run-error" if you exceed this limit.
Number of processes	You are not supposed to create multiple processes (threads). This is to no avail anyway, because your program has exactly 1 processor fully at its disposal. To increase stability of the jury system, there is a maximum of 10 processes that can be run simultaneously (100 for Java) (including processes that started your program). People who have never programmed with multiple processes (or have never heard of "threads") do not have to worry: a normal program runs in one process.

5.5 Java class naming

Compilation of Java sources is somewhat complicated by the class naming conventions used: there is no fixed entry point; any class can contain a method `main`. Furthermore, a class declared `public` must be located in an indentically named file.

In Domjudge, this is worked around by autodetecting the main class.

6 Printing

Printing is disabled for this contest. We also ask you not to print directly from your workstation.

7 Problem statements and support material

The problem statements and supporting material are available on the "Problem statements" tab of your team page. For each problem, we provide you with the problem statement as an HTML page, a PDF, as well as a ZIP archive containing the sample input and output. There is also a PDF file containing all problems, and a ZIP archive containing all sample input.

8 Compiling and testing your solution

You should use the following command lines to compile your program (replacing the bold parts as appropriate):

C	gcc -Wall -O2 -pipe -o **prob** **prob**.c -lm -lgmp
C++	g++ -Wall -O2 -pipe -o **prob** **prob**.cpp -lgmp -lgmpxx
Java	javac -d . **Prob**.java

And the following to test your solution with the sample input:

C	./**prob** < *.in | diff - *.out
C++	./**prob** < *.in | diff - *.out
Java	java -Xrs -Xss8m -Xmx98304k **Prob** < *.in | diff - *.out

9 Ranking

Teams are ranked by the total number of problems solved, where more points is better. This is the first number displayed in the "score" column of the scoreboard.

If teams tie after applying the previous rule, they are ranked by total number of penalities, where less penalities is better. For each problem that was solved on the nth attempt and t minutes after the start of the contest, you will receive $20 * (n-1) + t$ penalties. These numbers are displayed in the second and third "score" column.

10 C/C++ reference and javadoc

During the contest, you are allowed to access reference documents for C, C++ and Java. These provide you with references of the standard library of the respective language, and the GNU multiple precision arithmetic library for C/C++. We have provided a link to these from the "problem statements" page. Alternatively you can type in the address `http://doc.hc2.ch/` directly in your web browser.

11 Code examples

Below are a few examples on how to read input and write output for a problem.

The examples are solutions for the following problem: the first line of the input contains the number of testcases. Then each testcase consists of a line containing a name (a single word) of at most 99 characters. For each testcase output the string "Hello <name>!" on a separate line.

Sample input and output for this problem:

Input	Output
3 world Jan SantaClaus	Hello world! Hello Jan! Hello SantaClaus!

Note that the number 3 on the first line indicates that 3 testcases follow.
A solution for this problem in C:

```
1  #include <stdio.h>
2
3  int main()
4  {
5          int i, ntests;
6          char name[100];
7
8          scanf("%d\n", &ntests);
9
10         for(i=0; i<ntests; i++) {
11                 scanf("%s\n", name);
12                 printf("Hello %s!\n", name);
13         }
14
15         return 0;
16 }
```

Notice the last `return 0;` to prevent a RUN-ERROR!

A solution in C++:

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
        int ntests;
        string name;

        cin >> ntests;
        for(int i = 0; i < ntests; i++) {
                cin >> name;
                cout << "Hello " << name << "!" << endl;
        }

        return 0;
}
```

A solution in Java:

```
import java.io.*;

class Main
{
        public static BufferedReader in;

        public static void main(String[] args) throws IOException
        {
                in = new BufferedReader(new InputStreamReader(System.in));

                int nTests = Integer.parseInt(in.readLine());

                for (int i = 0; i < nTests; i++) {
                        String name = in.readLine();
                        System.out.println("Hello "+name+"!");
                }
        }
}
```

(You might want to take a look at java.util.StringTokenizer to parse lines containing multiple numbers.)

11.1 Using the GMP library

In C/C++ you may use the GNU Multiple Precision Arithmetic Library. Here follows a short example where two numbers a and b are read from standard input, and $res = a * b^{-1} + b \pmod{2^{32} + 15}$ is output.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <gmp.h>
4
5  int main(void)
6  {
7   mpz_t a, b, p, res;
8
9   /* Initialize memory: you have to do this once per mpz_t */
10  mpz_init(a);
11  mpz_init(b);
12  mpz_init(p);
13  mpz_init(res);
14
15  /* read a and b from stdin */
16  gmp_scanf("%Zd %Zd", a, b);
17
18  /* set p from string */
19  mpz_set_str(p, "4294967311", 10);
20
21  /* compute a * b^-1 + b mod p */
22  mpz_invert(res, b, p); /* res = invert b mod p */
23  mpz_mul(res, a, res); /* res = a * res */
24  mpz_add(res, b, res); /* res = b + res */
25  mpz_mod(res, res, p); /* res = res mod p */
26
27  /* Print res */
28  gmp_printf("%Zd\n", res);
29
30  /* free used memory */
31  mpz_clear(a);
32  mpz_clear(b);
33  mpz_clear(p);
34  mpz_clear(res);
35
36  return EXIT_SUCCESS;
37 }
```

You will find more details in the GMP documentation, which is available at `http://doc.hc2.ch`. Don't forget to add the compile flag `-lgmp`!