Amsterdam Algorithm Programming Preliminaries Rulebook 2017

STORM

September 20, 2017

Contents

Chapter 1 Definitions

AAPP: The Amsterdam Algorithm Programming Preliminaries 2017, also referred to as the contest. The contest is organised by STORM. It will take place on September 23th, 2017.

BAPC: The Benelux Algorithm Programming Contest Finals 2017 in Amsterdam, organised by study association via (University of Amsterdam). It will take place on October 7th, 2017.

NWERC: The Northwestern Europe Regional Contest Finale 2017 at Bath, United Kingdom. It will take place in the weekend of November 24th, 2017.

STORM: Study association for Mathematics and Computer Sciences studies, at the Vrije Universiteit Amsterdam.

Organisation: The members of the organising committee of STORM, also called AAPPCie.

Jury: The group of people responsible for checking the solutions submitted by the participants.

Tech: The group of people responsible for the system.

Balloon girls: Runners who are responsible for delivering print-outs, answering questions and awarding balloons to teams when a submission is correct.

Crew: Organisation, members of the jury, tech and balloon girls.

Participant: Member of a participating team that competes in the contest.

Submission: The submission of a solution by a team, which can be handed in using DOMjudge and will be checked by our servers.

Chapter 2 Organisation

2.1 The organisation consists of members of STORM.

2.2 The organisation has the right to stop the contest, extend the contest time, temporarily block submissions for all teams or change the scores in exceptional conditions.

2.3 In situations to which no rule applies, the organisation decides.

2.4 The jury consists of two jurors to be appointed by the organisation.

2.5 The organisation has formed the Tech, consisting of students of the Vrije Universiteit Amsterdam.

2.6 The organisation will appoint balloon girls who will watch over the contest areas during the contest, hand out the print-outs and balloons and will be available for practical questions during the contest.

2.7 It is possible for members of the organisation to participate the contest. In that case, the organisation will ensure that this participant will not break section **4.7.2** and section **4.7.3**. This means that by section **4.2.4**, the chairman of the organisation can not participate in the contest.

2.8 It is not possible for balloon girls, jury members and tech to participate in the contest, due to the fact that they are helping during the contest.

2.9 All crew members will be recognizable by their shirt.

Chapter 3 Participation

3.1 Introduction

3.11 Participation is only possible in student teams of up to 3 persons, see also section **2.7** if this person is a organisation member.

3.12 Changing the composition of a team is only possible with written permission of the organisation.

3.13 The organisation decides how many teams are allowed to compete.

3.14 The organisation has the right to deny the participation of teams before the start of the contest.

3.2 Student teams

A student team:

3.1.1 may participate for free.

3.1.2 consists of students from the Vrije Universiteit Amsterdam and who are not participating in any other team.

3.1.3 participates in the student teams pool for the title 'Winners of the Amsterdam Algorithm Programming Preliminaries 2017'.

3.1.4 consists of students who are eligible by the Eligibility Decision Tree (see also appendix), to participate in the student teams pool for a place at BAPC.

3.1.5 has exactly three students which satisfy section **3.1.4**, to participate in the student teams pool for a place at NWERC.

Chapter 4 The Contest

4.1 Introduction

4.1.1 The language used during the contest is English.

4.1.2 The contest lasts for 5 hours.

4.1.3 From the beginning until one hour before the end of the contest, the scores are displayed in DOMjudge.

4.1.4 In the last hour, no balloons will be handed out and participants can only see their own score in DOMjudge.

4.1.5 The final scoreboard will be shown at the presentation after the contest.

4.1.6 If a participant wants to leave the workplace during the contest, for example to use the bathroom or to smoke outside the building, a balloon girl has to accompany him or her.

4.2 Problems

4.2.1 The jury will provide at least 8 and at most 12 problems.

4.2.2 When a problem is unclear a 'clarification request' can be sent to the jury. The jury will respond to this request. If the response is relevant to all teams, the jury will send the response to all teams.

4.2.3 The jury has the right to change or withdraw problems during the contest. When this happens the jury will inform all teams.

4.2.4 Before the start of the contest, the content of the problem sets of the AAPP 2017 and their solutions are known only by the chairman of the organization, the jury and the tech.

4.3 Workplace

4.3.1 A workplace will be available for each team and all workplaces will be equal in equipment. The following equipment will be provided:

- One computer, with a screen, a mouse and a QWERTY keyboard.

- Input data on the computer itself or on the server, see also section **4.4.2**.
- Paper & pens.
- Three times the problem set of the AAPP 2017 which can be opened when the contest has started, see also section **4.7.1**.
- A copy of Rules Amsterdam Algorithm Programming Preliminaries 2017, including the DOMjudge Team Manual.

4.3.2 A team is allowed to bring up to 25 A4-sized pages, printed one-sided or up to 12 A4- sized pages, printed two-sided, of documentation. Each team member is allowed one identical copy.

4.3.3 A team is allowed to bring a dictionary; English to their native language.

4.3.4 You may bring mascots such as stuffed toy animals, as long as it does not violate section **4.5.2**, section **4.5.4** and section **4.5.5**. The mascot needs to be checked before the contest by a crew member.

4.4 System

4.4.1 A solution has to be written in

- C++
- Java
- Python

unless the problem statement explicitly states otherwise.

4.4.2 Input data is provided on your computer. This will be provided once and it will not be uploaded again during the contest.

4.4.3 The organisation expects that the teams have read the DOMjudge Team Manual before entering the contest and therefore have the knowledge to be able to hand in a submission, see also appendix for the manual.

4.4.4 The jury decides per programming language which libraries and function calls are allowed to be used in the solutions.

4.4.5 All prints made by the teams will be brought by a balloon babe. Participants are not allowed near the printers.

4.4.6 A team is not allowed to bring software.

4.5 House rules

4.5.1 The house rules apply to everybody inside the building.

4.5.2 The use of hardware, including all calculators, which are not approved by the organisation are strictly forbidden, with exception of simple watches and medical equipment.

4.5.3 The use of mobile phones, tablets, smart watches or any other electronic device during the contest is strictly forbidden. The organisation will provide safe storage for these devices during the contest.

4.5.4 Changing of hardware or operating software is strictly forbidden.

4.5.5 During the contest, communication within the team and crew is allowed. Communication with everyone else is forbidden during the contest.

4.5.6 Participants must follow any instructions given by the crew.

4.5.7 Participants will wear the shirt and badge provided by the organisation.

4.6 Judgement

4.6.1 Each submission is acknowledged.

4.6.2 For each problem, the jury has a correct solution and test data.

4.6.3 A submission is correct when it has a solution to the input in a time limit decided by the jury and the output is the same as the output of the jury (unless the problem statement explicitly states otherwise). This time limit is not announced to the teams.

4.6.4 The winner of a pool is decided by (in order):

(a) The team with the most correctly solved problems.

(b) The team with the least solving time. This is the sum of the time needed for each solved problem (defined as the time between the beginning of the contest and the submission of the first correct solution), plus a 20-minute penalty for each incorrect submission until the first correct submission. (Incorrect solutions for which a team has not submitted a correct solution or incorrect solutions submitted after a correct solution was accepted do not add to the solving time.)

(c) The team that first submitted its last accepted problem is ranked higher. In case a tie still remains, the team that first submitted its second-last accepted problem is ranked higher, and so on. In the event that this does not resolve the tie, the ranks will be determined by chance.

4.6.5 The jury is responsible for everything that has to do with the problem set and can be contacted for this through the 'clarification requests'.

4.7 Disqualification

The organisation has the right to disqualify teams for misbehavior/breaking the rules and can use section **2.3** for this. The organisation can disqualify a team among other reasons if the organisation thinks that a participant:

4.7.1 has opened the problem set before the contest started.

4.7.2 had access to the problem set before the contest started.

4.7.3 had access to the solutions before the contest started.

4.7.4 does not stick to the House Rules (section **4.5.1**, section **4.5.2**, section **4.5.3**, section **4.5.4**, section **4.5.5**, section**4.5.6** and section **4.5.7**).

4.7.5 did upload software on the computer, against section **4.4.6**.

4.7.6 did deliberately break section **4.1.6**.

Chapter 5 Prizes

The organisation and the main sponsor provided the following prize for the participants:

5.1 A trophy for the 'Winners of the Amsterdam Algorithm Programming Preliminaries 2017', for the participants in the student teams pool.

5.2 A silver medal for the 'Runner-up of the Amsterdam Algorithm Programming Preliminaries 2017', for the participants in the student teams pool.

5.3 A bronze medal for the 'Third-place finisher of the Amsterdam Algorithm Programming Preliminaries 2017', for the participants in the student teams pool.

5.4 At least two places for the BAPC finals, depending on the number of places the organisation of BAPC provides, if the team satisfies section **3.1.4**. Transport will be reasonably paid for by the organisation.

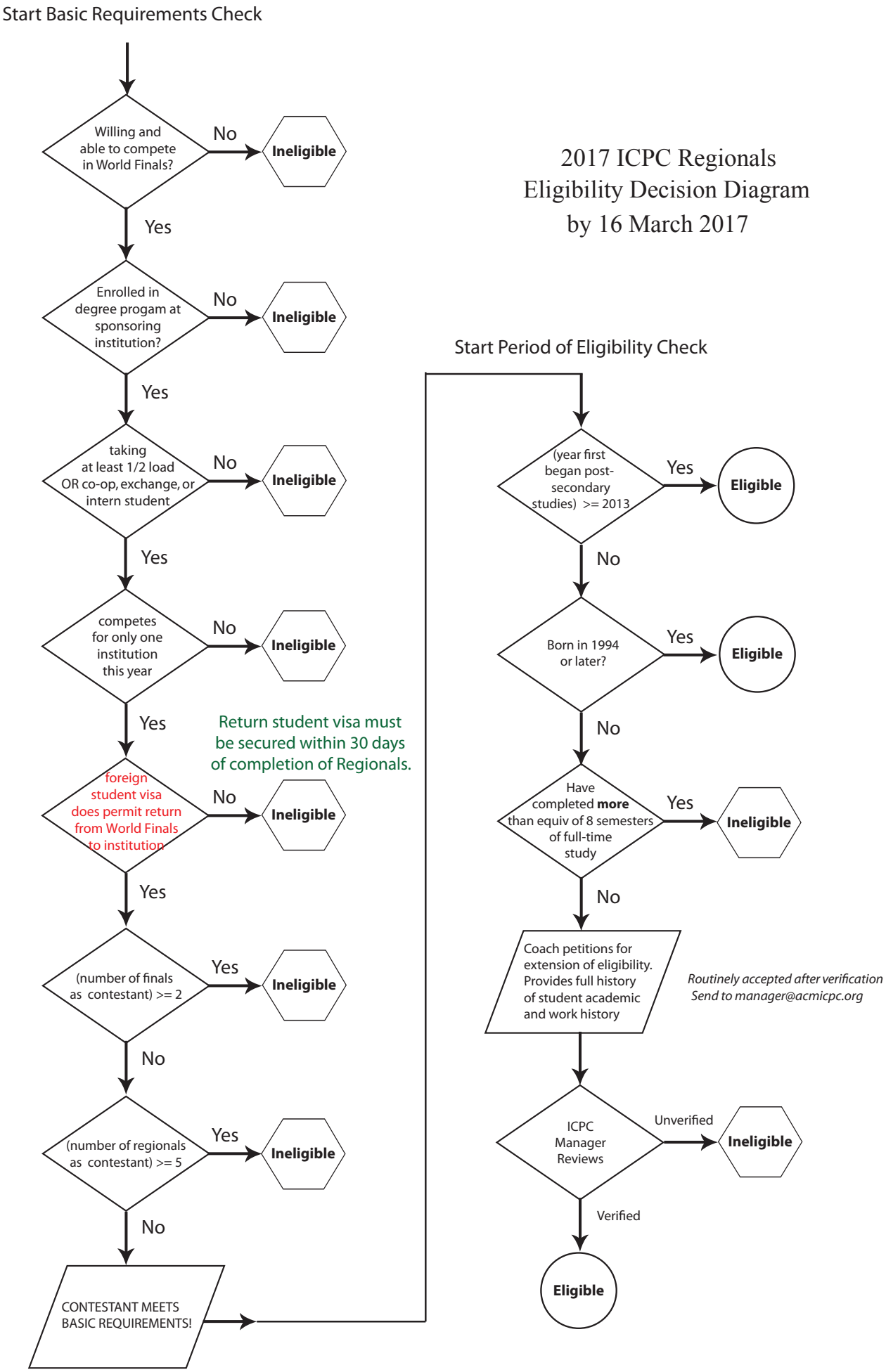
5.5 At least two places for the NWERC depending on the number of places the organisation of NWERC provides, if the team satisfies section **3.1.5**. NWERC teams are selected by the scores on BAPC. Transport and accommodation will be reasonably paid for by the organisation, up to a certain amount.

Chapter 6 Appendix

See next pages for the

- Eligibility Decision Tree
- Domjudge Team Manual including a NWERC scoreboard example

2017 ICPC Regionals
Eligibility Decision Diagram
by 16 March 2017



This page is intentionally left blank.

DOMjudge team manual

Summary

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 at least one of your team's members reads all of this manual. There are specific details of this jury system that might become of importance when you run into problems. **BE WARNED!**

DOMjudge works through a web interface that can be found at `http://example.com/domjudge/team`. See figures 1 and 2 on the next page 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 appendix A for some code examples.

Submitting solutions

You can submit solutions with the command-line program `submit` or from the web interface:

Command-line
: Use `submit <filename>`. If your filename is of the form `<problem>.<extension>` where `<problem>` is the label of the problem and `<extension>` is a standard extension for your language, then these will automatically be detected. For a complete reference of all options and examples, see `submit --help`.

Web interface
: From your team page, `http://example.com/domjudge/team`, click the file selection button in the left column and select the file(s) you want to submit. By default, the problem is selected from the base of the (first) filename and the language from the extension.

Viewing scores, submissions, etc.

Viewing scores, submissions and sending and reading clarification requests and replies is done through the web interface at `http://example.com/domjudge/team`.

End of summary

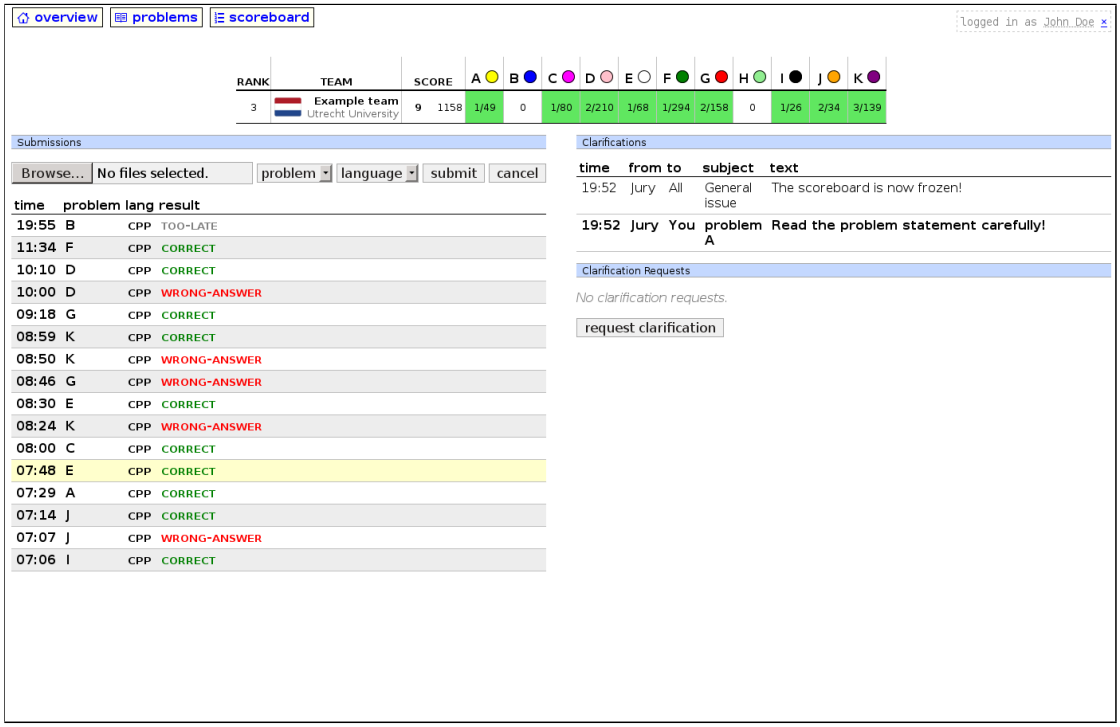


Figure 1: the team web interface overview page.

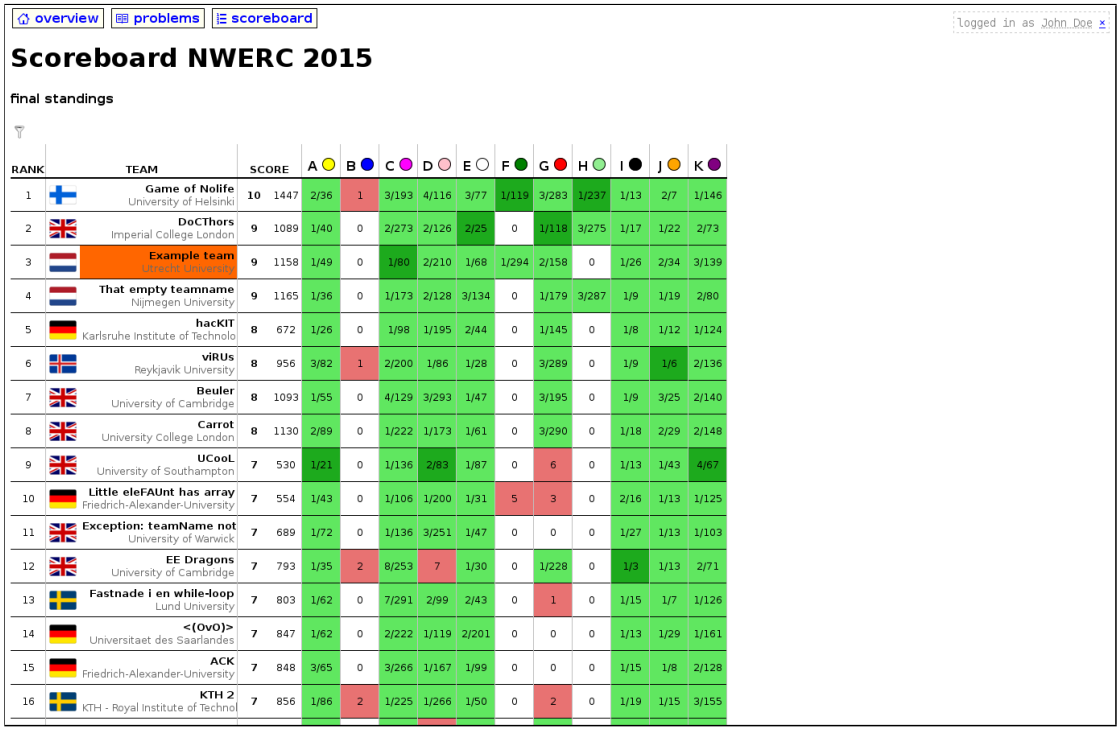


Figure 2: the scoreboard webpage.

Version/revision: 5.1.3 / ffbe9c1
Last modified: February 12, 2017
Generated: February 12, 2017

1 Submitting solutions

Submitting solutions can be done in two ways: with the command-line program `submit` or using the web interface. One of the interfaces might not be available, depending on the system configuration by the jury. A description of both methods follows.

1.1 Command-line: `submit`

Syntax: `submit [options] filename.ext ...`

The submit program takes the name (label) of the problem from `filename` and the programming language from the extension `ext`. This can be overruled with the options `-p problemname` and `-l languageextension`. See `submit --help` for a complete list of all options, extensions and some examples. Use `submit --help | more` when the help text does not fit on one screen.

`submit` will check your file and warns you for some problems: for example when the file has not been modified for a long time or when it's larger than the maximum source code size. Filenames must start with an alphanumerical character and may contain only alphanumerical characters and `+._-`. You can specify multiple files to be part of this submission (see section 4 'How are submissions being judged?').

Then `submit` displays a summary with all details of your submission and asks for confirmation. Check whether you are submitting the right file for the right problem and language and press 'y' to confirm. `submit` will report a successful submission or give an error message otherwise.

The submit program uses a directory `.domjudge` in the home directory of your account where it stores temporary files for submission and also a log file `submit.log`. Do not remove or change this directory, otherwise the `submit` program might fail to function correctly.

1.2 Web interface

Solutions can be submitted from the web interface at http://example.com/domjudge/team. In the left column click the file selection button and select one or multiple files for submission. DOMjudge will try to determine the problem and language from the base and extension of the (first) filename. Otherwise, select the appropriate values. Filenames must start with an alphanumerical character and may contain only alphanumerical characters and `+._-`.

After you hit the submit button and confirm the 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.

2 Viewing the results of submissions

The left column of your team web page shows an overview of your submissions. It contains all relevant information: submission time, programming language, problem and status. The address of your team page is http://example.com/domjudge/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 scores of all teams. Many cells will show additional "title text" information when hovering over them. The score column lists the number of solved problems and the total penalty time. Each cell in a problem column lists the number of submissions, and if the problem was solved, the time of the first correct submission in minutes since contest start. This is included in your total time together with any penalty time incurred for previous incorrect submissions. Optionally the scoreboard can be 'frozen' some time before the end of the contest. The full scoreboard view will not be updated anymore, but your team row will. Your team's rank will be displayed as '?'. Finally, via the top menu you can also view the list of problems and view/download problem texts and sample data, if provided by the jury.

2.1 Possible results

A submission can have the following results (not all of these may be available depending on configuration of the system):

CORRECT	The submission 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 (this option might be disabled).
TIMELIMIT	Your program took longer than 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 than the limit, etc. Also check that your program exits with exit code 0!
NO-OUTPUT	Your program did not generate any output. Check that you write to standard out.
OUTPUT-LIMIT	Your program generated more output than the allowed limit. The output was truncated and considered incorrect.
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.
TOO-LATE	Bummer, you submitted after the contest ended! Your submission is stored but will not be processed anymore.

3 Clarifications

All communication with the jury is to be done through 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; you can associate a specific problem or one of the general categories to a request. This clarification request is only readable for the jury. The jury can answer specifically to your team or send a reply to everyone if it is relevant for all.

4 How are submissions being judged?

The DOMjudge jury system is fully automated. In principle no human interaction is necessary. The judging is done in the following way:

4.1 Submitting solutions

With the `submit` program or the web interface (see section 1) you can submit a solution to a problem to the jury. Note that you have to submit the source code of your program (and not a compiled program or the output of your program).

On the jury system your program enters a queue, awaiting compilation, execution and testing on one of the jury computers.

4.2 Compilation

Your program will be compiled on a jury computer running Linux. All submitted source files will be passed to the compiler which generates a single program to run; for languages where this is relevant, the first file specified will be considered the 'main' source file.

Using a different compiler or operating system than the jury should not be a problem. Be careful however, not to use any special compiler and/or system specific things (you may be able to check compiler errors on the team page).

The jury system defines `ONLINE_JUDGE` and `DOMJUDGE`. These are defined as preprocessor symbols in compiled languages and as (environment) variables in scripted languages.

4.3 Testing

After your program has compiled successfully it will be executed and its output compared to the output of the jury. Before comparing the output, the exit status of your program is checked: if your program gives the correct answer, but exits with a non-zero exit code, the result will be a RUN-ERROR! There are some restrictions during execution. If your program violates these it will also be aborted with a RUN-ERROR, see section 4.4.

When comparing program output, it has to exactly match to output of the jury, except that some extra whitespace may be ignored (this depends on the jury's configuration of the problems). So take care that you follow the output specifications. In case of problem statements which do not have unique output (e.g. with floating point answers), the jury may use a modified comparison function.

4.4 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 total amount of source code in a single submission may not exceed 256 kilobytes, otherwise your submission will be rejected.

memory During execution of your program, there are 524288 kilobytes of memory available. This is the total amount of memory (including program code, statically and dynamically defined variables, stack, Java VM, ...)! If your program tries to use more memory, it will abort, resulting in a run error.

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 core fully at its disposal. To increase stability of the jury system, there is a maximum of 15 processes that can be run simultaneously (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.

4.5 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 the default configuration of DOMjudge this is worked around by autodetecting the main class. When this feature is not used, then the main class should be "`Main`", with method "`public static void main(String args[])`", see also the Java code example in appendix A.

Version/revision: 5.1.3 / ffbe9c1
Last modified: February 12, 2017
Generated: February 12, 2017

A 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++:

```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  int main()
7  {
8          int ntests;
9          string name;
10
11         cin >> ntests;
12         for(int i = 0; i < ntests; i++) {
13                 cin >> name;
14                 cout << "Hello " << name << "!" << endl;
15         }
16
17         return 0;
18 }
```

A solution in Java:

```
1  import java.io.*;
2
3  class Main
4  {
5          public static BufferedReader in;
6
7          public static void main(String[] args) throws IOException
8          {
9                  in = new BufferedReader(new InputStreamReader(System.in));
10
11                 int nTests = Integer.parseInt(in.readLine());
12
13                 for (int i = 0; i < nTests; i++) {
14                         String name = in.readLine();
15                         System.out.println("Hello "+name+"!");
16                 }
17         }
18 }
```

A solution in C#:

```
using System;

public class Hello
{
        public static void Main(string[] args)
        {
                int nTests = int.Parse(Console.ReadLine());

                for (int i = 0; i < nTests; i++) {
                        string name = Console.ReadLine();
                        Console.WriteLine("Hello "+name+"!");
                }
        }
}
```

A solution in Pascal:

```
program example(input, output);

var
        ntests, test : integer;
        name         : string[100];

begin
        readln(ntests);

        for test := 1 to ntests do
        begin
                readln(name);
                writeln('Hello ', name, '!');
        end;
end.
```

And finally a solution in Haskell:

```
import Prelude

main :: IO ()
main = do input <- getContents
          putStr.unlines.map (\x -> "Hello " ++ x ++ "!").tail.lines $ input
```