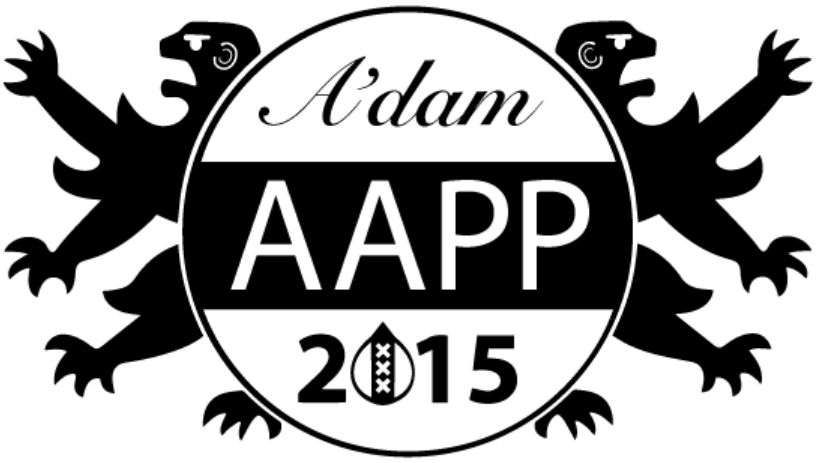


AAPCie Draaiboek

STORM

November 28, 2015

Contents

Chapter 1 Definitions

AAPP: The Amsterdam Algorithm Programming Preliminaries, also referred as the contest, is a programming contest for VU students. The contest is organised by studyassociation STORM. It takes place mid/end september. AAPP are the preliminaries of BAPC.

BAPC: The Benelux Algorithm Programming Contest, organised by a studyassociation in the Benelux. It takes place in the end of october. Since 2016, BAPC are the preliminaries of NWERC instead of AAPP.

NWERC: The Northwestern Europe Regional Contest. It takes place in the end of november. NWERC are the preliminaries of ICPC World Finales.

ICPC World Finales The International Collegiate Programming Contest (ICPC) World Finales are held around march.

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

Website: Will be maintained by the organisation and contains information about problems from last year and the rules of AAPP. The website is availible at `http://www.storm.vu/aapp`.

Jury: Members of the organisation, who are responsible to check the submission of the contestants.

Tech: Members of the organisation, who are responsible for the system.

Balloon girls: Runners who are responsible for handing out balloons after a submission is correct, for handing out prints and for answering your questions during the contest.

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

Deelnemers: Members of team, who are joining the contest.

Submission: The submission of a team, which can be hand in via DomJudge and will be check by our own servers.

Chapter 2 How does a programming contest works?

2.1 Preliminaries

For students of the VU Amsterdam, the International Collegiate Programming Contest (ICPC) knows four rounds:

1. Amsterdam Algorithm Programming Preliminaries (AAPP): Open for all VU Amsterdam teams which satifsy the conditions, see also the Appendix 7.3. The top three teams are allowed to BAPC.

2. Benelux Algorithm Programming Contest (BAPC): The top three teams of each educational institution can accompete to BAPC. The top two teams of each educational institution are allowed to NWERC, provided that the team consists three team members (see also Appendix 7.3).

3. Northwestern Europe Regional Contest (NWERC): The top two teams of each educational institution can accompete to NWERC. The top three teams of each educational institution are allowed to the ICPC World Finales.

4. International Collegiate Programming Contest (ICPC) World Finales: The top three teams of the region Northwestern Europe are allowed to the ICPC World Finals.

Teams that do not satisfy the Eligibility Decision Tree (see appendix 7.3), are usually allowed as spectator if there is room for them.

2.2 General Concept of the Contest

Each team (with a maximum of three students) is trying to solve as many problems (most of the time between the 10 and the 13) as possible in five hours, by programming a program on the computer. The program reads the

input from the input file, search or computes the right answer and gives back the results as output.

Most of the time, the problems are based on known classic algorithms, like the shortest path algorithm of Dijkstra or backtracking. Often there are a number of mathematical problems.

Logically, may only discuss with their teammates. Furthermore, it is allowed to use a cheat sheet (Team Reference Document) and to bring their own keyboard. Teams that have a problem correctly within four hours, gets a balloon of a balloon babe in the color of the problem.

2.3 Sending in a Submissions

Via the jury interface DomJudge (a website), teams can hand in the submissions. Further access to the internet has been blocked during the contest. The jury interface also tells the teams, if the submissins was correct or not. The program will be rated on the following two criteria: accuracy and efficiency.

That means: the program has solve a set of test cases (not only the one given as input in the problem) in previously set time limit (most of the time a few secondes).

The following reactions of the jury are possible:

- Accepted;
- Wrong Answer;
- Timelimit Exceeded;
- Runtime Error.

For each correct solution (Accepted), you will get a balloon from a balloon babe (if the correct solution was hand in within the first four hours).

2.4 Program languages

The following program languages are accepted at the contest:

AAPP	Java, C, C++, C++11, C#, Haskell
BAPC	Java, C, C++
NWERC	Java, C, C++
ICPC World Finales	Java, C, C++

2.5 Score

The team score depends on two parts:

- The number of correct solved problems within the five hours
- The total time (including the penalty time).

There are two parts per solved problem:

- The numbers of minutes since the start of the contest and the moment till solving the problem
- 20 penalty time for each wrong solution

A wrong solution only gives penalty time if the problem is solved later the contest.

On the scoreboard, teams will be sorted by the numbers of solved problems. Teams who has same amount of solved problems, will be sorted on total time. In the last hour of the contest, the scoreboard will be freezed. You will see from your own team if a problem is correct or not, but it is not possible to see on the scoreboard if others teams have hand in a (correct) solution.

See for more information the rulebook, Appendix 7.5 (Judgement).

2.6 Time Schedule

Each program contest has the same set up when its come to time. Some organisations decided to spread this event over a weekend, to create some room for excursions and sponsoring activities. Most of the time, this is the case at NWERC, since teams are coming from far. However, teams likely do not like it that it takes that long before they can really start with the contest.

The time schedule for the contest is as follow:

Registration Registration of the incoming teams. The organisation gives teams and coaches goodiebags and sponsored T-shirts, which we are obligatory to wear during the day(s). At the registation, it is also possible that the organisation wants to check the cheatsheets and the keyboards (30 minuten).

Welcome Speech Teams worden verwelkomt door de voorzitter en de hoofdsponsor. Uitleg van de spelregels, het doornemen van de dagplanning en een praatje van de hoofdsponsor (maximaal 30 minuten).

Testsessie	Voor de contest is er een testsessie. Deze testsessie wordt gebruikt om te kijken of de wedstrijdomgeving aan alle verwachtingen voldoet (maximaal 1 uur).
Coach meeting	Meeting voor de coach, zie ook sectie 3.8.2 (maximaal 30 minuten).
Lunch	Meestal verzorgd door de hoofdsponsor (1 uur).
Last remarks	Laatste uitleg, vragen of teams nog brandende vragen hebben (hooguit een kwartiertje, maar er wordt 30 minuten gerekend zodat iedereen tijd heeft om op zijn plek te gaan zitten voor de contest).
Contest	Contest begint (5 uur).
Freeze scoreboard	Scoreboard staat op freeze, men kan alleen hun eigen inzendingen zien en of deze goed zijn of niet (4 uur na dat de contest is begonnen).
Borrel	Borrelen (1 uur)
Prijsuitreiking	Uiteindelijke scoreboard wordt gepresenteerd (10 minuten)
Presentatie problems	Presentatie met de oplossingen wordt gepresenteerd door de jury (15 minuten).
Dinner	Optioneel: hangt er vanaf of de commissie geld heeft om het eten te vergoeden voor de teams.

Chapter 3 To-Do - Crew

3.1 Chairman

De voorzitter leidt de commissie en zorgt ervoor dat alles in goede banen loopt. Hij/zij delegeert en motiveert de leden om hun taken uit te voeren (zie ook sectie 3.1.1). Tevens leidt hij/zij de vergaderingen (zie ook sectie 3.1.1).

3.1.1 Tasks

Leading the Meetings

Vergader regelmatig, maar alleen als het ook nuttig en nodig is voor de commissie. Zorg ervoor dat iedereen de kans krijgt om iets in te brengen tijdens de vergadering.

TIMELINE 1: *Vergadering plannen 101*

Dag 0	Bedenken of je volgende week tijd heb om te vergaderen
Dag 1	Vergaderplanner sturen + actiepunten
Dag 5	Reminder: planner invullen
Dag 6	Notulen doorlezen of aan de notulist vragen of ze af zijn
Dag 6	Agenda opstellen
Dag 7	Mailen: datum vergadering + notulen + agenda + actiepunten
Dag 7	Reserveren: vergaderzaal
Dag 8 - 14	Printen: notulen + agenda
Dag 8 - 14	Reminder: vergadering
Dag 8 - 14	Vergaderen!
Ooit	Wachten op de (klad)notulen

Hier is een opzet voor de agenda:

1. Opening
2. Agenda
3. Notulen Vorige Vergadering (NVV)
4. Post, Email en Mededelingen (PEM)
5. Oude Actiepunten
6. {*kies een onderwerp*}
7. {*kies een onderwerp*}
8. Wat Verder Ter Tafel Komt (WVTTK)
9. Samenwerkingsrondje
10. Datum Volgende Vergadering (DVV)
11. Sluiting

Aansturen van de leden

Motiveren

Het enthousiast kunnen motiveren en coachen van leden, kan ervoor zorgen dat men graag willen blijven helpen binnen de commissie. Om er voor te zorgen dat men gemotiveerd blijft, moet er een duidelijk doel zijn. Niet alleen het doel (het organiseren van de contest), maar ook de actiepunten moeten duidelijk zijn anders worden deze niet uitgevoerd.

Verder helpt het om leden eens in de zoveel tijd te herinneren aan hun taken of actiepunten, en om (vriendelijk) te vragen of ze deze al hebben uitgevoerd. Het helpt ook om te vragen of ze hulp van andere leden nodig hebben om de taak uit te voeren.

Verbondenheid met de commissie zorgt er ook voor dat een lid meer zin krijgt om hun werk uit te voeren. Samen eten voor de vergadering, taart tijdens de vergadering en na de vergadering samen aan de slag gaan in de Stuka wil wel werken.

Speeches houden

3.1.2 Time Schedule

3.1.3 Tips

3.2 Secretaris

Mail bijhouden

Binnenkomende mail labelen en verwerken:

1. Doorsturen naar de juiste persoon of personen
2. Beantwoorden
3. Archiveren
4. Verwijderen

3.3 Penningmeester

Financiën bijhouden

Begroting

Afrekening

Subsidie

Bijzitter

3.3.1 Contest Director

Voorzitter van de contest, bepaald samen met de jury en de tech of de contest kan beginnen en maakt besluiten over de contest. Draagt verantwoordelijkheid voor de contest.

3.4 Sponsoring

3.5 Tech

3.6 Jury

3.7 Balloon babes

Als teams een opdracht goed hebben binnen vier uur, krijgen ze een ballon van een balloon babe. Deze schoonheid zal deze ballonen netjes vastzetten aan het bureau of aan een bureaustoel. Naast deze zware taak, doen ze ook het volgende voor de wedstrijd:

- Helpen bij de registratiebalie
- Controleren dat niemand begint, voordat de contest is begonnen
- Controleren dat niemand de opgave opent, voordat de contest is begonnen
- Mobieltjes laten inleveren.

En tijdens de wedstrijd voeren ze de volgende taken uit:

- Ballonnen uitdelen

- Printjes aangeven
- Meelopen (roken, wc, eten)
- Controleren dat men niet vals speelt.

3.8 Coaches

Coach zijn van een team bij BAPC of NWERC, houdt in dat je het team ondersteunt daar waar nodig is en om de commissie te vertegenwoordigen bij eventuele coachmeetings

3.8.1 Voorbereiding

Ondersteuning geven aan een team houdt in dat je het onderstaande zal moeten regelen of voorbereiden:

- Vervoer naar de wedstrijd (auto, OV etcetera)
- De registratie, zodat men kan deelnemen aan de competitie. Sowieso invoeren in ICPC, maar soms zijn er ook aanvullende formulieren (the local registration form).
- Koffie vinden
- Aanmelden bij de registratie balie. Vergeet dan niet de onderstaande informatie bij de hand te hebben:
  - Teamnamen
  - Totaal aantal personen inclusief coach(es), voor het aantal goodiebags
  - Aantal mannen, aantal vrouwen inclusief coach(es), voor het geval dat er wel vrouwen T-shirts zijn.
  - T-shirtmaten (voor vrouwen geldt dat als het unisex/mannen T-shirts zijn, een maat kleiner nemen dan normaal gesproken wordt gekozen)
- Cheatsheets uitprinten/regelen
- Training regelen of zorgen dat ze zelf trainen
- Het team eraan herinneren dat je je eigen toetsenbord mee mag nemen

Tip: het kan handig zijn om een Whatsapp groep te starten met de teamleden en de coaches erin.

Nadat de wedstrijd is begonnen, mag de coach zich niet meer op de contest ground bevinden. **Tip: neem wat mee om de vijf uur door te komen.**

3.8.2 Coach meeting

Meestal wordt er tijdens de test sessie of tijdens de wedstrijd, een coach meeting gehouden om eventuele problemen over de wedstrijd zelf te kunnen bespreken. Omdat je de commissie/universiteit vertegenwoordigd, is het handig om met de commissie de onderstaande vragen te bespreken voor vertrek. De volgende vragen kunnen daarnaast nog besproken worden:

- Welke universiteit organiseert de volgende editie(s) van de wedstrijd?
- Hoe zijn de voorrondes gegaan?

Chapter 4 To-Do AAPP

4.1 Algemeen

Hieronder vindt je alle taken die moeten worden uitgevoerd, die niet te maken hebben met sponsoring of het technische deel van de wedstrijd.

4.1.1 Registratie starten

4.1.2 Promotie

Maandelijks mailing

Posters

Onderwijscoördinatoren laten mailen naar studie maillijst

4.1.3 Contact teams

Deelnemers informeren

Week voor de contest begint

Deelnemers informatie verwerken

ICPC & Google Drive

Evaluatie

Data sturen

4.1.4 Kleding

Crew

Deelnemers

4.1.5 VU Diensten

IT

STORM en IT hebben goede banden met elkaar. Hierdoor is het mogelijk om spullen te lenen voor de contest, mits we deze netjes terugbrengen:

- Computers
  Vraag rond juni aan IT S&E Werkplek Ondersteuning om 25 pc's vrij te houden voor de "rekenwedstrijd". Waarschijnlijk heeft Willem hier al eerder aan gedacht dan wij.
- Printer(s)
- Kooikar(ren)
- Monitorkabels
- Kluis
- Ricoh papier sleutel

FCO

VU busje

4.1.6 Bedankjes

IT

Willem houdt van slagroomtaart en Emilio houdt van alles waar chocolade in zit of wat met liefde is gemaakt.

Jury

Spelletjes

4.2 Sponsoring

4.2.1 Locatie (hoofdsponsor

4.2.2 Prijzen

4.2.3 Goodiebags

4.2.4 Bedrijventeams

4.2.5 Overige reclame

4.3 Tech

4.3.1 Cliënt

4.3.2 DOMJudge

4.3.3 DOMjura

4.3.4 Printen

4.3.5 Data opslaan

4.3.6 Puppet

4.4 Inpaklijst

Op één of andere manier vergeten we op de dag zelf altijd wat.. Hier moet dus een inpaklijst komen te staan. Of ergens in de Appendix zodat je alleen de losse pagina hoeft uit te printen.

Chapter 5 One week before the contest

5.1 Dag voor de contest

5.1.1 Printen

- ☐ De opgaven
- ☐ Het regelement
- ☐ Excelsheet met teamnamen, deelnemers, maten en bijzonderheden (toetsenbord, cheatsheet)

Chapter 6 Ideeën

- Volgende keer alle compilers testen op de server en clients!
- In de inschrijf form apart voor- en achternaam.
- Duidelijkheid over alcoholgebruik.
- Zelfde shirt als via?
- Proberen de opgaven/oplossingen/testdata eerder te krijgen van landelijke BAPC commissie.
- Bedankjes voor de jury op de dag van de contest.
- - DOMjudge in de vakantie maken
- Draaiboeken schrijven - technisch kant
- Image bewaren
- Eerstejaars programmeer wedstrijd
- ICPC - adres gegevens zijn nodig
- T-shirts zijn fijner dan polo's
- Duidelijker formulier - volledig namen graag
- Geen Python meer
- Ondersteunen qua talen alleen wat NWERC ondersteund
- Extra scherm inpakken
- Extra stekkerdozen inpakken
- Pas als de jury akkoord geeft, dan pas balloon geven. Balloon interface geeft eerder een balloon dan dat de jury correct geeft.

- DOMjura
- 2x printers
- Balloon babes inlichten over taken
- Cheatsheet controleren
- Prijzen: steam tegoed. Rasberry pi
- Training geven (Madelon, Renske)

Chapter 7 Appendix

7.1 Wall of Fame: Crew

7.1.1 Organisation 2014-2015

Committee

De commissie bestond uit de volgende mensen:

7.1.2 Organisation 2013-2014

Committee

Het samenwerkingsverband UvA-VU leverde de volgende samenstelling op:

- Renske Augustijn (Voorzitter, VU)
- Mylène Martodihardjo (Vice-voorzitter, VU)
- Bram van den Akker (Sponsoring, UvA)
- Daniel Maaskant (Sponsoring, VU)
- Michael Vasseur (System Director, VU)
- Tirza Jochemsen (VU)
- Ruben Helsloot (VU)
- Iris Meerman (UvA)
- Bas van den Heuvel (UvA)

Balloon babes

- Mylène Martodihardjo (Chief balloon babe)
- Sherida van den Bent (VU)
- Nicolette Stassen (VU)
- Saskia Kreuzen (VU)
- Ysbrand Galama (UvA)

Jury

- Frank Blom (Head judge)
- Alex ten Brink (Jurymember of BAPC 2014 finale)

7.1.3 Organisation 2012-2013

Committee

There was not really a committee that year, but the following members did help to set up the preliminaries:

- Mylène Martodihardjo (algemeen, fotograaf)
- Michael Vasseur (tech, fotograaf)
- Mark Laagland (tech)
- Jip de Beer (sponsoring)
- Kylie van de Moot (fotograaf)

Balloon babes

- Mylène Martodihardjo
- Kylie van de Moot

Jury

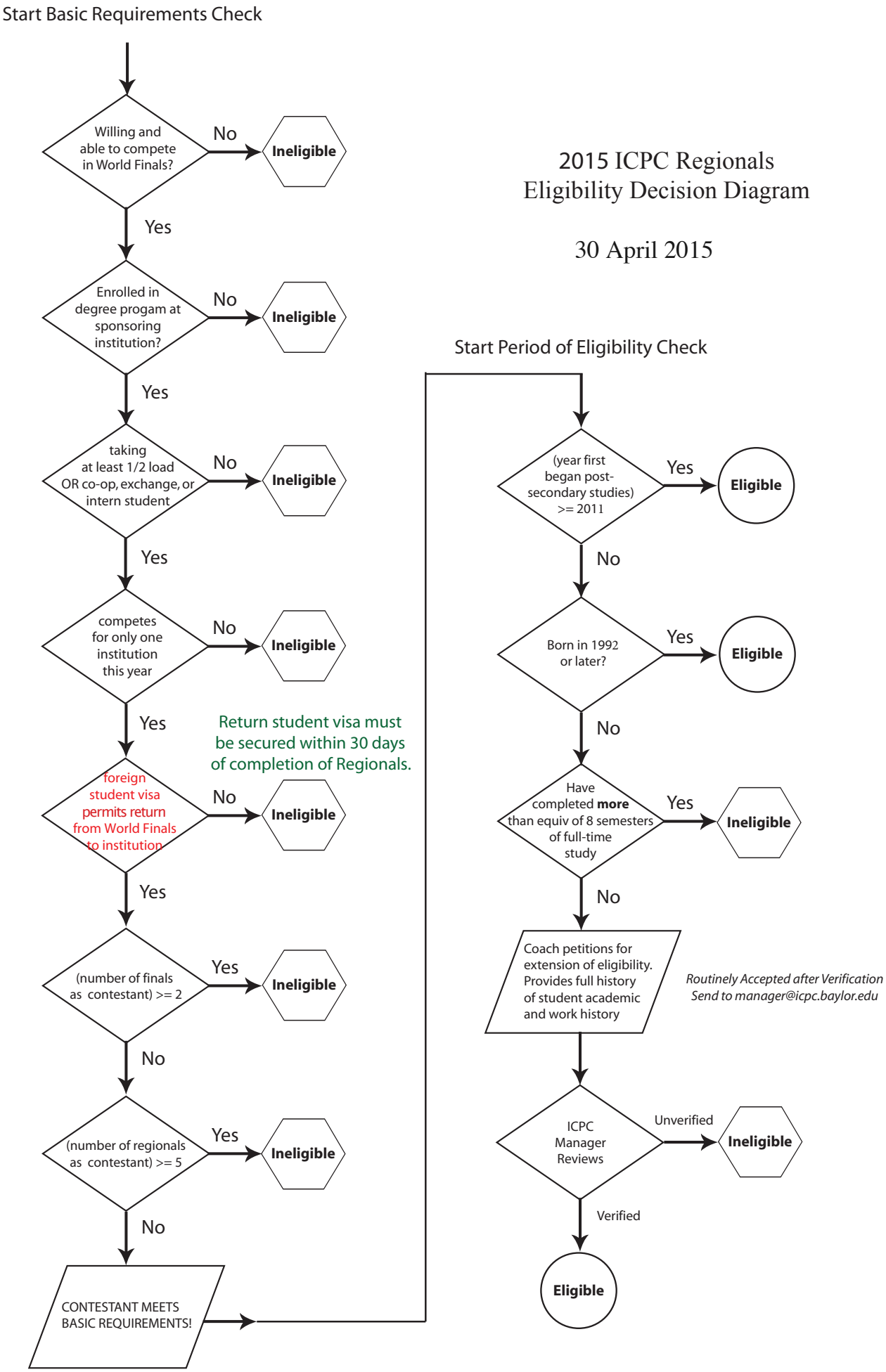
- Michael Vasseur
- Mark Laagland

7.2 Winnaars voorrondes

7.3 Eligibility Decision Tree

2015 ICPC Regionals
Eligibility Decision Diagram

30 April 2015



This page is intentionally left blank.

7.4 DOMjudge team manual

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 examples.

Submitting solutions

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

Command-line
: Use `submit <problem>.<extension>`, where `<problem>` is the label of the problem and `<extension>` is a standard extension for your language. For a complete reference of all options and examples, see `submit --help`.

Web interface
: From your team page, http://example.com/domjudge/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. Click **Add another file** to add more files to the submission.

Viewing scores, submissions, etc.

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

End of summary

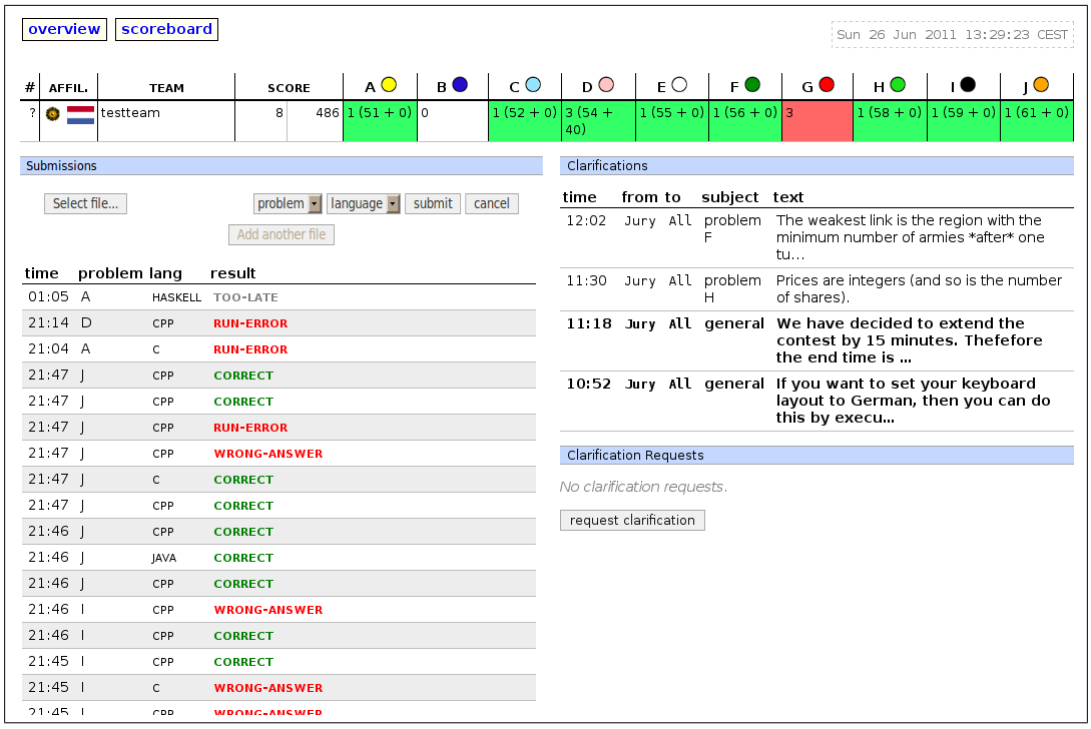


Figure 1: the team web interface overview page.

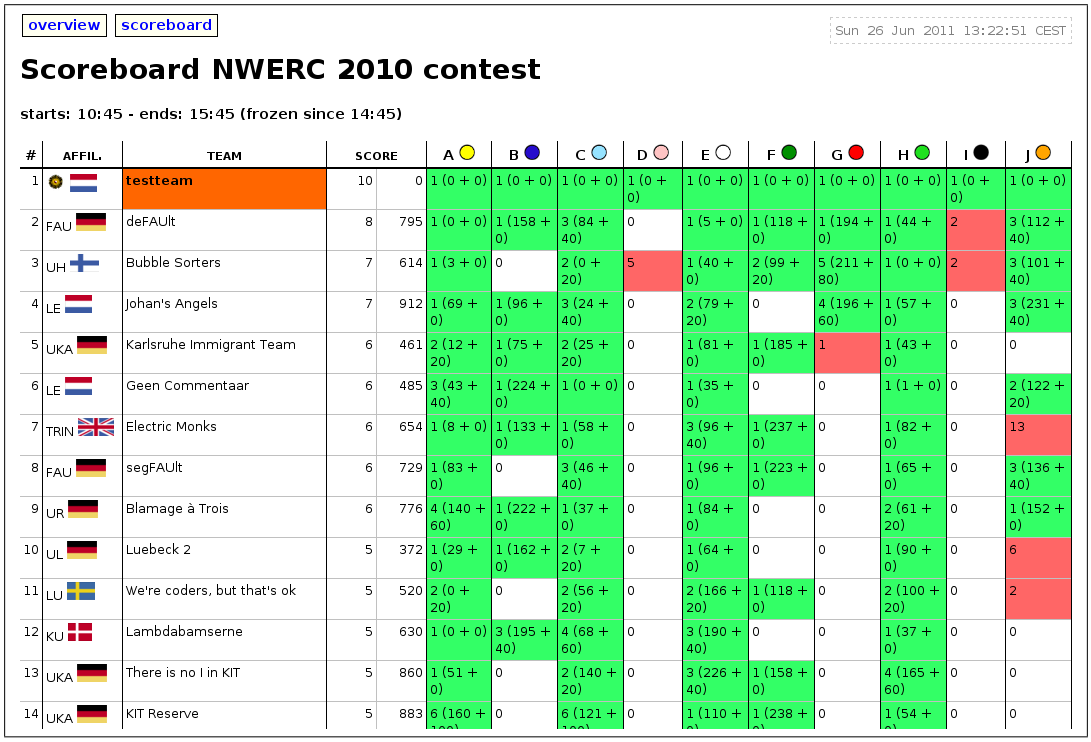


Figure 2: the scoreboard webpage.

Version/revision: 4.0.5 / 4db4d42
Last modified: March 23, 2015
Generated: March 23, 2015

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 **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. Filenames must start with an alphanumerical character and may contain only alphanumerical characters and `+._-`.

When you've selected the first source file, you may use the **Add more files** button to specify additional files to be part of this submission (see section 4 'How are submissions being judged?').

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 '?'.

2.1 Possible results

A submission can have the following results:

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.

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. This result is optional and might be disabled.

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 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.

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).

There 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 out of them; for languages where that is relevant, the first specified file 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 gecompiled 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. 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 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.

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#:

```
1  using System;
2
3  public class Hello
4  {
5          public static void Main(string[] args)
6          {
7                  int nTests = int.Parse(Console.ReadLine());
8
9                  for (int i = 0; i < nTests; i++) {
10                         string name = Console.ReadLine();
11                         Console.WriteLine("Hello "+name+"!");
12                 }
13         }
14 }
```

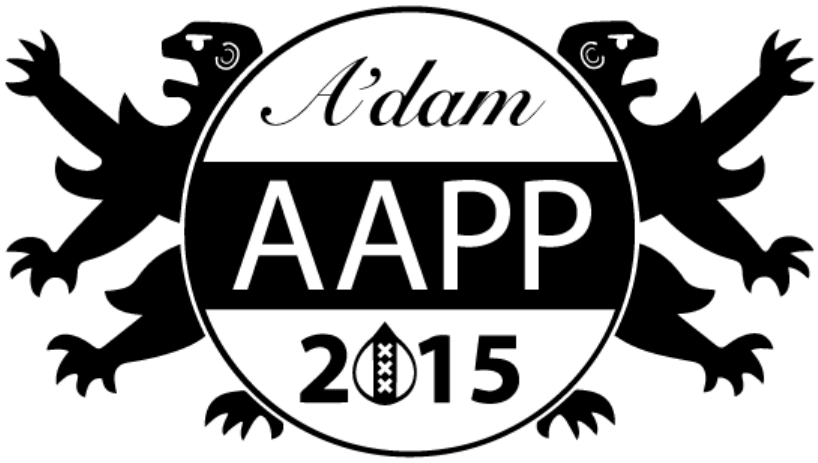
A solution in Pascal:

```
1  program example(input, output);
2
3  var
4          ntests, test : integer;
5          name         : string[100];
6
7  begin
8          readln(ntests);
9
10         for test := 1 to ntests do
11         begin
12                 readln(name);
13                 writeln('Hello ', name, '!');
14         end;
15 end.
```

And finally a solution in Haskell:

```
1  import Prelude
2
3  main :: IO ()
4  main = do input <- getContents
5            putStr.unlines.map (\x -> "Hello " ++ x ++ "!").tail.lines $ input
```

7.5 Amsterdam Algorithm Programming Preliminaries



Amsterdam Algorithm Programming Preliminaries Rulebook 2015

STORM

September 16, 2015

Contents

Chapter 1 Definitions

AAPP: The Amsterdam Algorithm Programming Preliminaries 2015, also referred to as the contest. The contest is organised by STORM. It will take place on September 19th, 2015.

BAPC: The Benelux Algorithm Programming Contest Finals 2015 in Leiden, organised by study association De Leidsche Flesch. It will take place on October 24th, 2015.

NWERC: The Northwestern Europe Regional Contest Finale 2015 at Linköping University, Sweden. It will take place in the weekend of November 28th, 2015.

STORM: Study association for Mathematics and Computer Sciences studies, at VU University in Amsterdam.

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

Website: Maintained by the organisation and among others provides information, problems from previous years and rules Amsterdam Algorithm Programming Preliminaries 2015. The website is available at `http://www.storm.vu/aapp`.

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 VU University in 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 VU University Amsterdam and who are not participating in any other team.

3.1.3 has a coach, which is the contact person of a team. This can be a team member or a student or staff member of the institution.

3.1.4 participates in the student teams pool for the title 'Winners of the Amsterdam Algorithm Programming Preliminaries 2015'.

3.1.5 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.

3.1.6 has exactly three students which satisfy section **3.1.5**, 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 and will be placed on the website.

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 2015 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 two screens, a mouse and a QWERTY keyboard.
- Input data on the computer itself, see also section **4.4.2**.
- Paper & pens.
- Three times the problem set of the AAPP 2015 which can be opened when the contest has started, see also section **4.7.1**.
- A copy of Rules Amsterdam Algorithm Programming Preliminaries 2015, 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
- C++
- C#
- Haskell
- Java
- Python

unless the problem statement explicitly states otherwise. Note that C# and Haskell are not allowed during the NWERC, and that Python is not allowed on BAPC nor NWERC.

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 (it is possible to win more than one price as a team) for the participants:

5.1 A trophy for the 'Winners of the Amsterdam Algorithm Programming Preliminaries 2015', for the participants in the student teams pool.

5.2 At least three places for the BAPC finals, depending on the number of places the organisation of BAPC provides, if the team satisfies section **3.1.5**. Transport will be reasonably paid for by the organisation.

5.3 At least two places for the NWERC depending on the number of places the organisation of NWERC provides, if the team satisfies section **3.1.6**. Transport and accommodation will be reasonably paid for by the organisation, up to a certain amount.

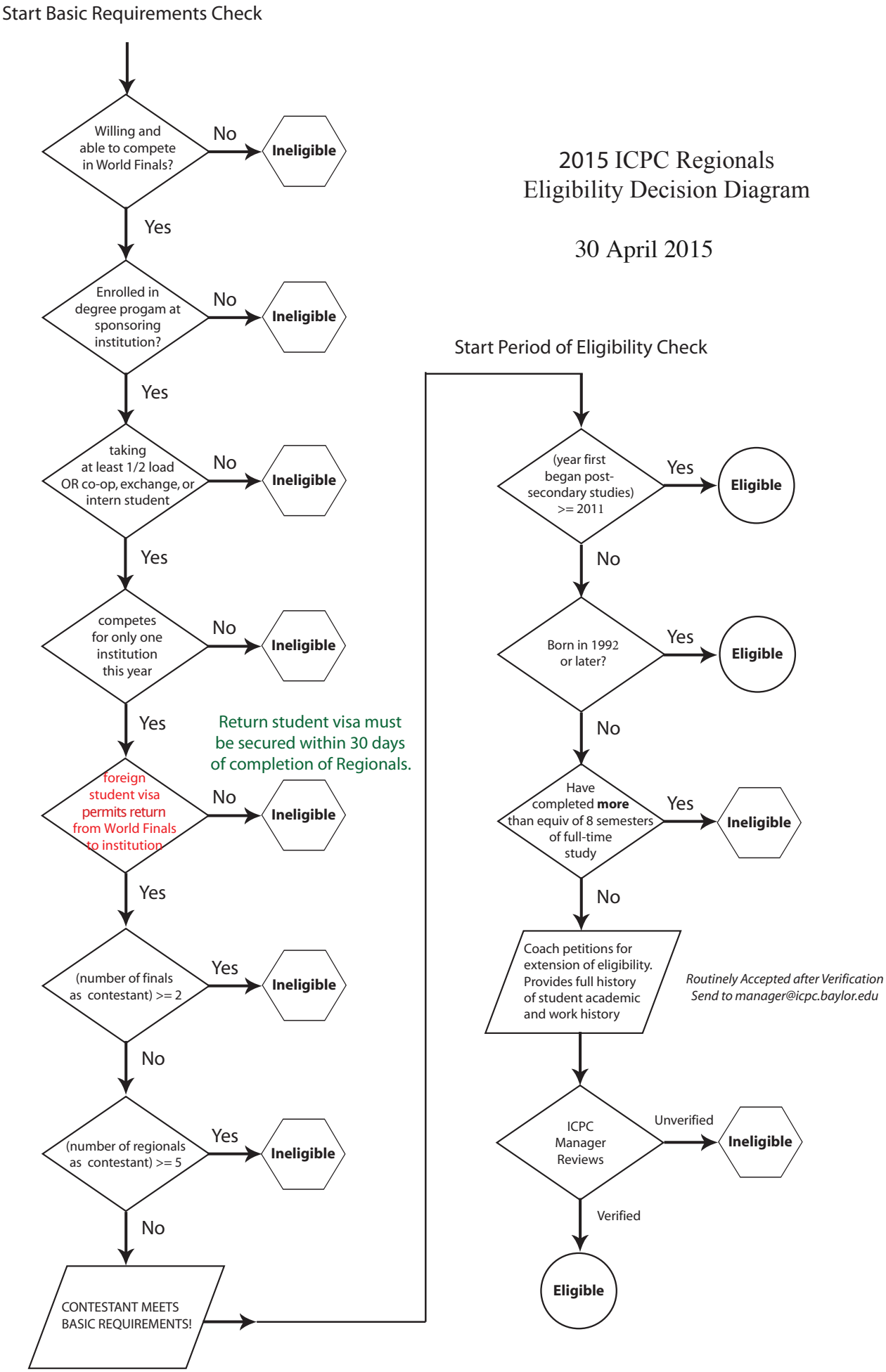
Chapter 6 Appendix

See next pages for the

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

2015 ICPC Regionals
Eligibility Decision Diagram

30 April 2015



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 examples.

Submitting solutions

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

Command-line
: Use `submit <problem>.<extension>`, where `<problem>` is the label of the problem and `<extension>` is a standard extension for your language. For a complete reference of all options and examples, see `submit --help`.

Web interface
: From your team page, http://example.com/domjudge/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. Click **Add another file** to add more files to the submission.

Viewing scores, submissions, etc.

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

End of summary

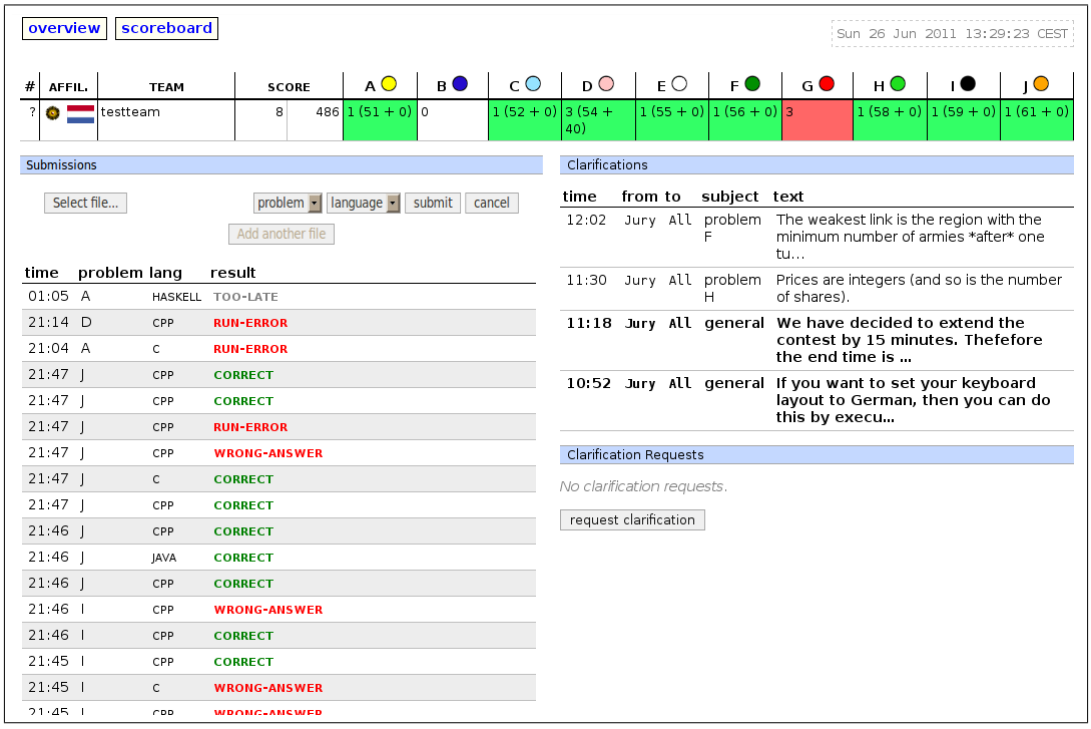


Figure 1: the team web interface overview page.

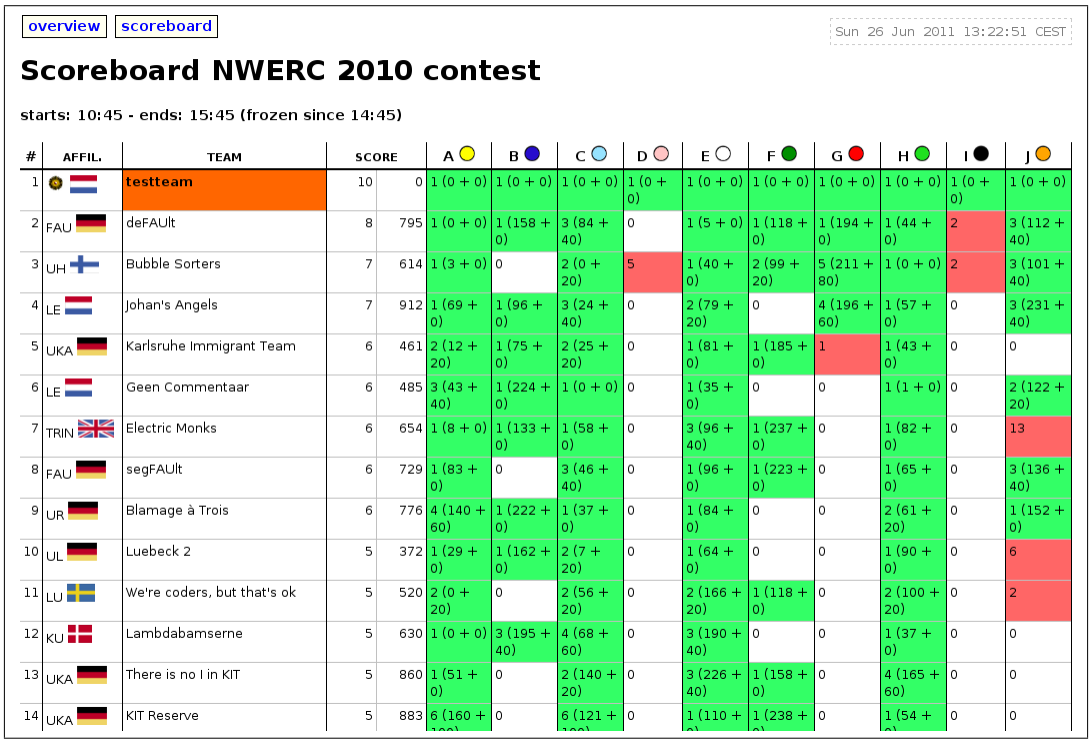


Figure 2: the scoreboard webpage.

Version/revision: 4.0.5 / 4db4d42
Last modified: March 23, 2015
Generated: March 23, 2015

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 **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. Filenames must start with an alphanumerical character and may contain only alphanumerical characters and `+._-`.

When you've selected the first source file, you may use the **Add more files** button to specify additional files to be part of this submission (see section 4 'How are submissions being judged?').

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 '?'.

2.1 Possible results

A submission can have the following results:

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.

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. This result is optional and might be disabled.

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 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.

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).

There 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 out of them; for languages where that is relevant, the first specified file 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 gecompiled 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. 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 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.

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
```