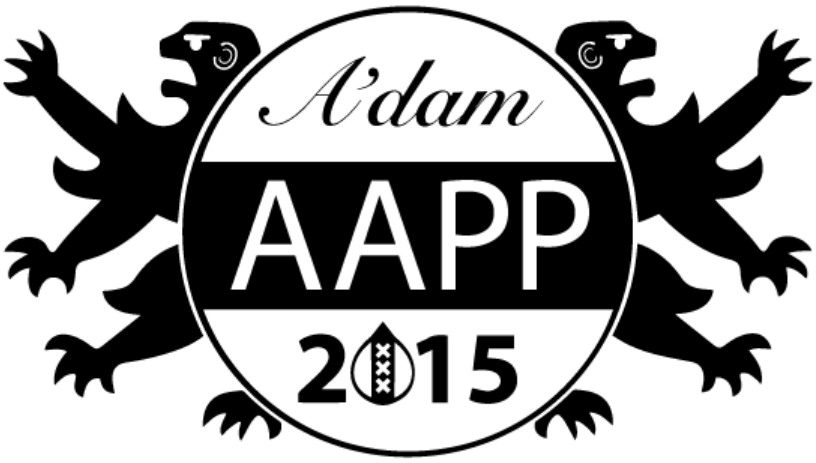


AAPCie Draaiboek

STORM

November 24, 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.

Dat wil zeggen: het programma moet een vooraf bepaalde set testcases (dat kan ook een case zijn die niet gegeven is als input in de problems) correct oplossen binnen een vooraf vastgestelde tijdslimiet (meestal enkele seconden).

De mogelijke reacties van de jury zijn onder andere:

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

Voor elke goede oplossing (Accepted) krijgt men een ballon van een balloon babe.

2.4 Programmeertalen

De onderstaande talen zijn toegestaan bij de genoemde contests:

AAPP Java, C, C++, C++11, C#, Haskell

BAPC Java, C, C++

NWERC Java, C, C++

ICPC World Finales Java, C, C++

2.5 Puntentelling

De score van een team bestaat uit twee onderdelen:

- Het aantal opgaven opgelost in vijf uur
- De totale tijd (de penalty time)

Per opgeloste opgave bestaat dat uit de volgende twee onderdelen:

- Het aantal minuten tussen het begin van de wedstrijd en het oplossen van de opgave
- 20 strafminuten voor elke foute inzending

Een foute inzending levert alleen strafminuten op als de opgave later alsnog wordt opgelost.

De teams worden gesorteerd op aantal opgeloste opgaven. Teams met evenveel opgaven worden gesorteerd op tijd. Het laatste uur wordt het scorebord bevroren (freeze). Je hoort dan nog wel of je eigen inzendingen goed of fout zijn, maar niet welke opgaven de andere teams nog oplossen.

Zie voor een uitgebreider uitleg het reglement, Appendix 7.5 (Judgement).

2.6 Tijdschema

Elke programmeerwedstrijd heeft dezelfde opzet qua tijdschema. Sommige kiezen er echter voor om deze indeling uit te rekken over een weekend, om zo meer ruimte te creëren voor excursies en sponsoring praatjes. Dit is meestal het geval bij NWERC, omdat deelnemende teams ook uit het buitenland komen. Door teams wordt het uitrekken van de planning vooral als vervelend ervaren, omdat het evenement veel langer duurt dan nodig is.

Het tijdschema ziet er als volgt uit:

Registratie Het registreren van teamsleden. Organisatie deelt de verplichte gesponserde T-shirts en geeft de eventuele goodiebags. Eventueel controleren en innemen van cheatsheets. Toetsenborden kunnen eventueel ook gecontroleerd worden (30 minuten).

Welkomswoord	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 Organisatie 2014-2015

7.1.2 Organisatie 2013-2014

Commissie

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 Organisatie 2012-2013

Commissie

Er was niet echt een commissie hiervoor, maar de onderstaande leden deden de organisatie

- 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

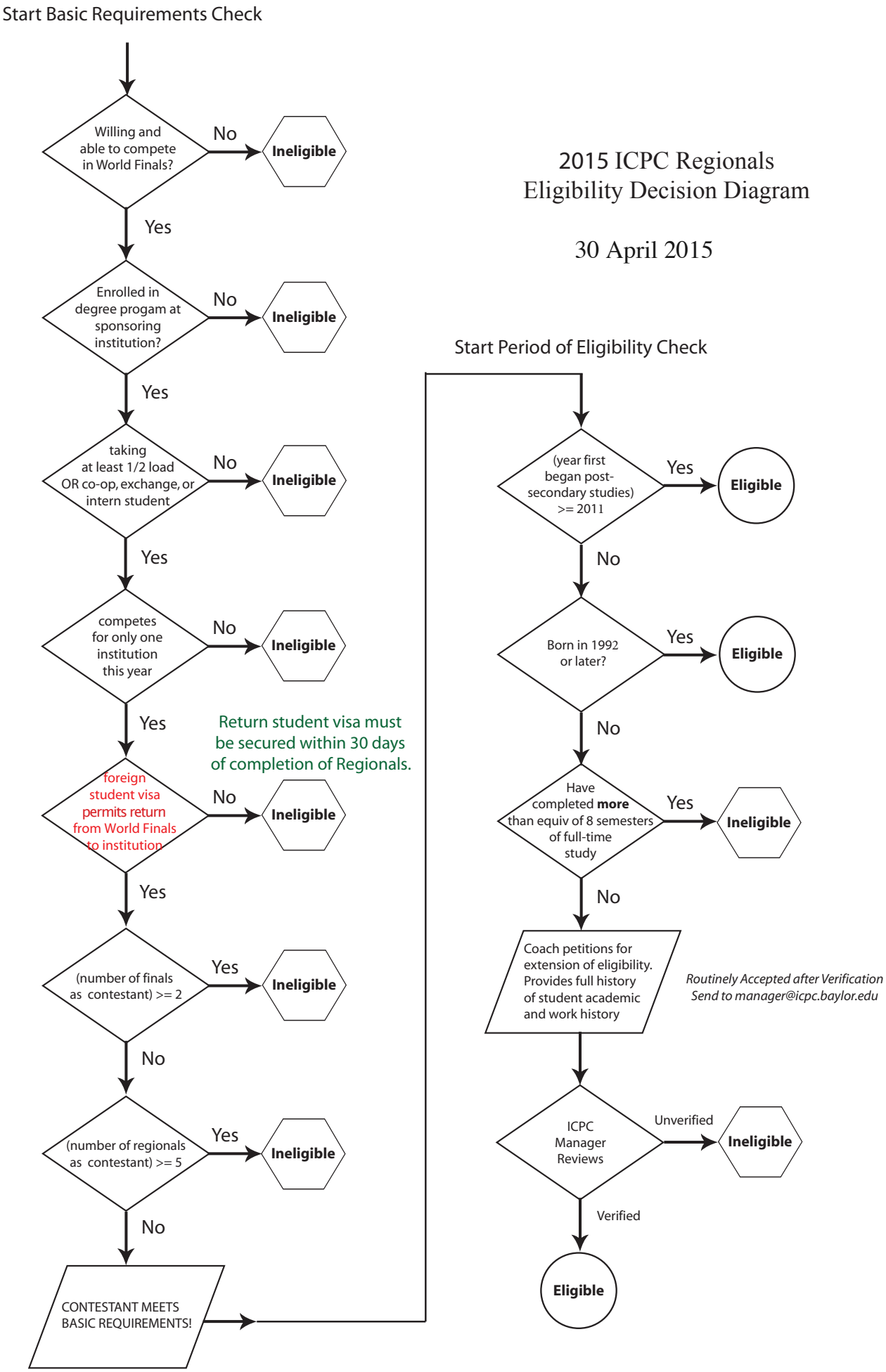
- Michael Vasseur
- Mark Laagland

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

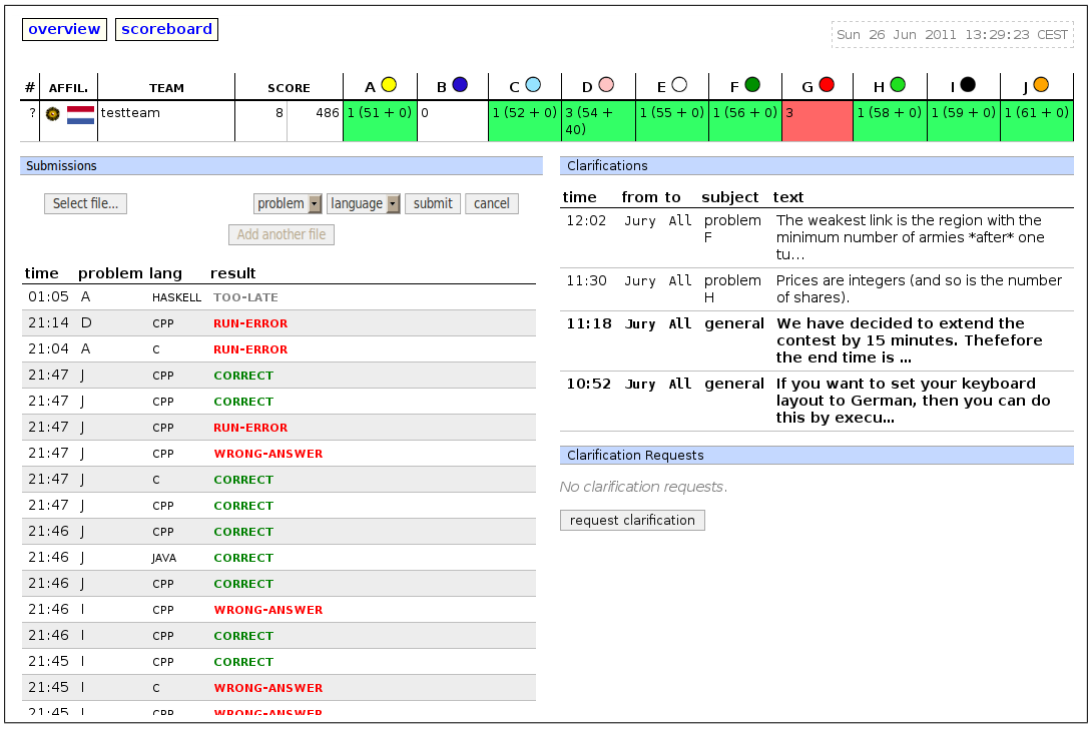


Figure 1: the team web interface overview page.

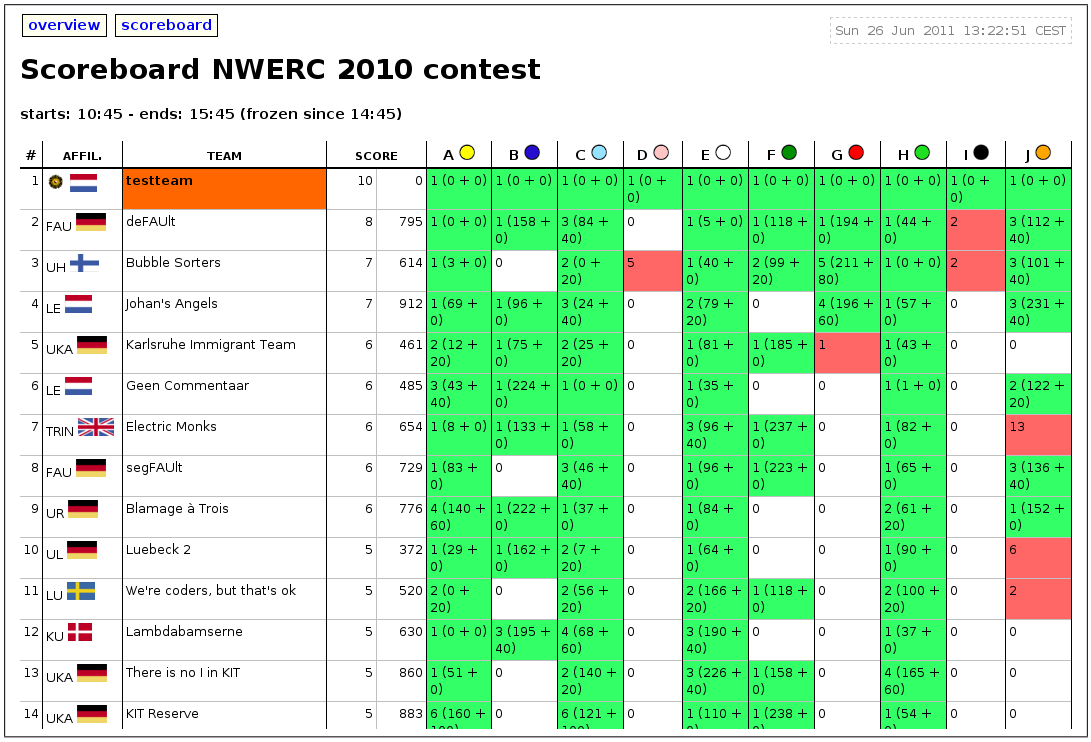


Figure 2: the scoreboard webpage.

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 }
```

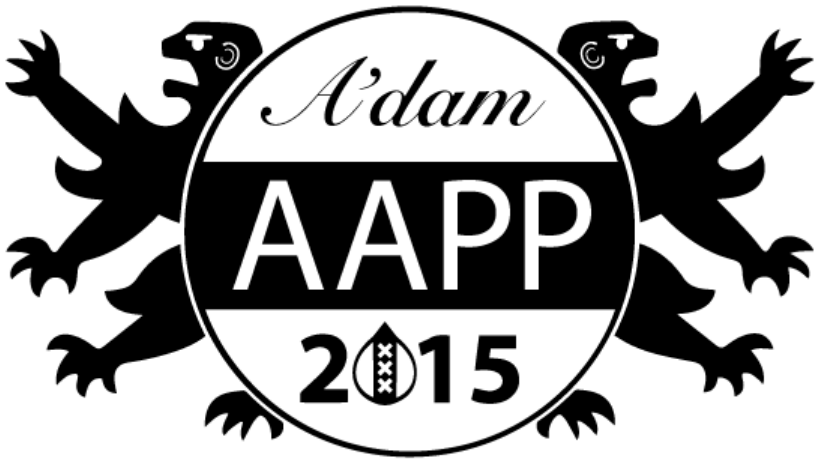
A solution in Pascal:

```
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.