DOMJUDGE TEAM MANUAL

(FOR THE ACM SOUTH PACIFIC PROGRAMMING CONTEST)

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.

DOMjudge works through a web interface – your documentation will give you the URL.

See Figure 4 on page 3 for an impression.

1 SUBMITTING SOLUTIONS

Solutions can be submitted from the web interface.

On the menu bar (see figure 1) click the green **Submit** button to open the submission dialogue box (see figure 2).

Browse to find your source code. Do NOT select an executable file or a project/solution file.

DOMjudge will try to determine the problem and language from the base and extension of the filename respectively. Otherwise, select the appropriate values from the drop down lists **problem** and **language**.

Note that the current compilers and versions on DOMjudge are: C/C++ – gcc version 8.3.0, Python 3.7.3, Java 11.0.4.

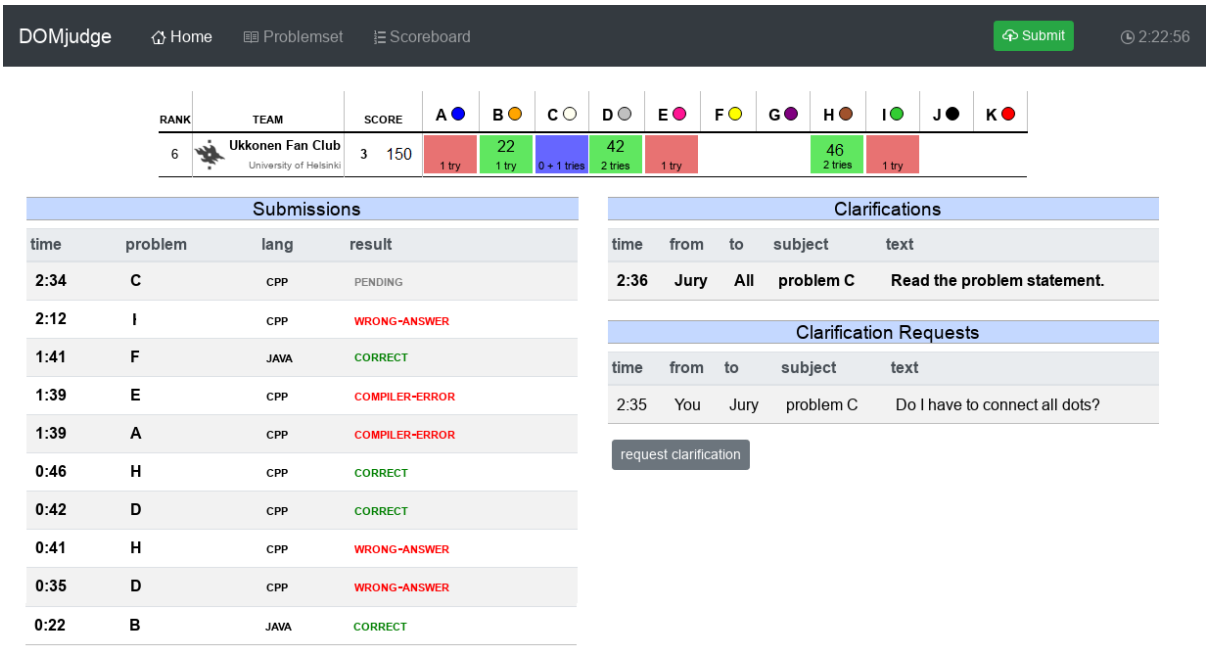

Figure 1 Overview

Figure 2 Making a submission

Compiler error messages will be provided to assist teams with different local environments.

After you hit the submit button and confirm the submission, you will be redirected back to your submission list page. On this page, 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 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 (see figure 4 on page 3). The scoreboard will be 'frozen' one hour before the end of the contest. The full scoreboard view will not be updated any more, but your team row will. Your team's rank will be displayed as `?'.

RANK	TEAM	SCORE	A	B	C	D	E	F	G	H	I	J	K
6	Ukkonen Fan Club University of Helsinki	3 150	1 try	22 1 try	0 + 1 tries	42 2 tries	1 try			46 2 tries	1 try		

Figure 3 Team scoreboard

ANZAC6 NZPC

RANK	TEAM	SCORE	A Parking [3 points]	B Arithmtc [3 points]	C Cooking [3 points]	D Croquet [3 points]	E Netball [10 points]	F Set [10 points]	G Postfix [10 points]	H Landing [10 points]	I-Mastermind [30 points]	J-GoingFishing [30 points]	K-SequencePeriods [30 points]	L-PasswordRequirements [30 points]	M-Surfing [100 points]	N-Parcels [100 points]
1	Self-Registered TaxiOverflow University of Melbourne	3121071	12 1 try	7 1 try	25 1 try	47 1 try	69 1 try	37 1 try	118 3 tries	161 2 tries	110 1 try		143 2 tries	3 tries	172 1 try	2 tries
2	MyFriendsArentHere University of New South Wales	3121649	3 1 try	9 1 try	22 1 try	63 1 try	74 1 try	80 1 try	103 3 tries	126 1 try	163 1 try		282 4 tries		261 5 tries	
3	minecraftbois University of New South Wales	2831614	295 2 tries	2 tries		1 try	289 1 try		289 1 try		37 1 try	255 2 tries		1 try	139 1 try	15 tries
4	LaserCorgis Royal Melbourne Institute of Technology	2821830	1 1 try	5 1 try	203 2 tries	15 2 tries	148 7 tries	76 1 try	101 2 tries	275 5 tries	147 3 tries				277 7 tries	
5	WindyDay Ateneo de Manila University	259 1143	166 1 try	175 2 tries	223 2 tries	1 try	2 tries	285 1 try	1 try	53 1 try	38 1 try			3 tries	95 1 try	7 tries
6	Shisuko Ateneo de Manila University	232 603	2 1 try	5 1 try	19 1 try	30 1 try	53 1 try	60 1 try	8 tries	2 tries			4 tries		268 2 tries	
7	EDIEEE University of New South Wales	229 771	9 1 try	17 1 try	100 4 tries			130 1 try	101 1 try	2 tries			1 try		118 3 tries	
8	uofa2	2191245	38 2 tries	159 1 try	209 6 tries		150 3 tries						4 tries		243 3 tries	
9	vinash Monash University	206 835	30 1 try	35 1 try	1 try			78 1 try			132 2 tries		237 1 try	127 3 tries		5 tries
10	tzha2101 The University of Sydney	190 860									46 1 try	220 6 tries	4 tries	198 1 try		
11	typedefl Massey University	172 975	5 1 try	11 1 try	28 1 try	57 1 try	147 1 try	133 1 try	4 tries	98 1 try	189 2 tries	3 tries	1 try			
12	ThreeNullPointers The University of Sydney	1521491	2 1 try	21 3 tries	65 3 tries	232 1 try	163 3 tries	223 1 try	181 1 try	214 2 tries			1 try	1 try		
13	gomango University of New South Wales	149 722	5 1 try	20 2 tries	43 3 tries				64 2 tries		289 1 try		1 try		201 2 tries	
14	nablahlv The University of Sydney	149 944	6 1 try	19 1 try	3 tries	121 1 try	111 1 try	143 1 try	145 2 tries	212 1 try		3 tries				7 tries

Figure 4 Public Scoreboard

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.
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 output.
WRONG-ANSWER	The output of your program was incorrect. This can happen simply because your solution is not correct, but remember that your output must comply exactly with the specifications of the jury.
TOO-LATE	Unfortunately you submitted after the contest ended! Your submission is stored but will not be processed any more.

3 CLARIFICATIONS

All communication with the jury is to be done with clarifications (see figure 5). 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.

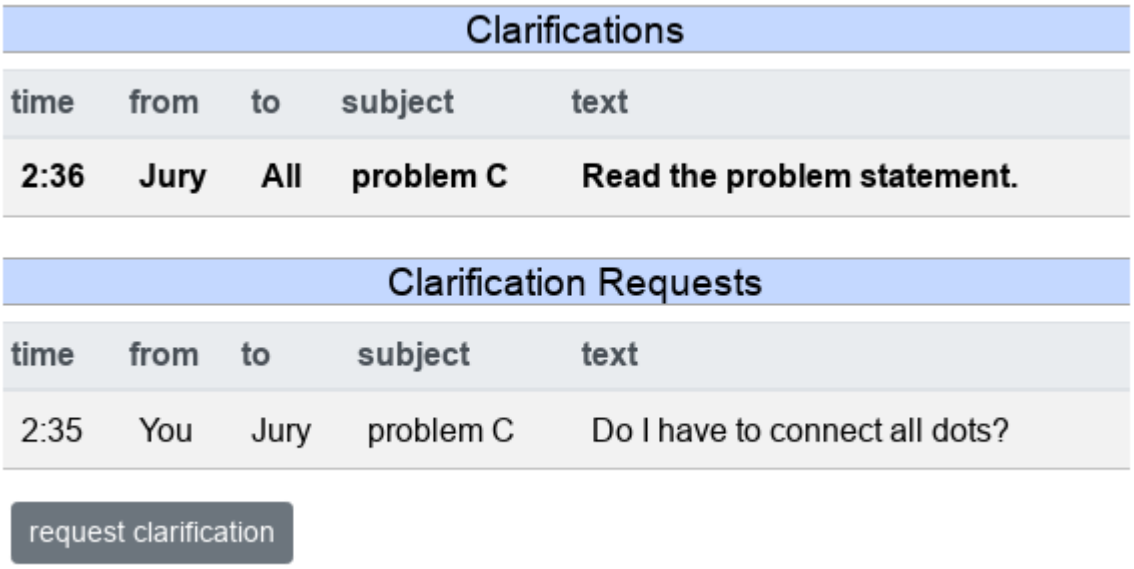

Figure 5 Clarifications

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

You submit a solution to a problem to the jury from the Web interface. 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. The submitted source file will be passed to the compiler which generates a program to run.

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 (eg Microsoft libraries for C++).

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! (see Restrictions below).

Java does not require an exit code. However, be careful about catching exceptions. If you handle all exceptions in your code, you may get WRONG-ANSWER instead of the potentially more useful RUN-ERROR.

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 a specified time. 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 2 gigabytes of memory available. This is the total amount of memory (including program code, statically and dynamically defined variables, stack, Java VM, etc)! 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.

APPENDIX : 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 test cases. Then each test case consists of a line containing a name (a single word) of at most 99 characters. For each test case output the string `Hello <name>!` on a separate line.

Sample input and output for this problem:

Sample Input	**Output for Sample Input**
3 world Jan SantaClaus	Hello world! Hello Jan! Hello SantaClaus!

Note that the number 3 on the first line indicates that 3 test cases follow.

1 Solutions for Contest Languages

i C:

```c
#include <stdio.h>

int main()
{

    int i, ntests;
    char name[100];

    scanf("%d\n", &ntests);

    for(i=0; i < ntests; i++)
    {
        scanf("%s\n", name);
        printf("Hello %s!\n", name);
    }
    return 0;
}
```

Notice the last line, `return 0`, which prevents a RUN-ERROR!

ii C++:

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

using namespace std;

int main()
{

    int ntests;
    string name;

    cin >> ntests;

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

Notice the last line, `return 0`, which prevents a RUN-ERROR!

iii Java:

```java
import java.io.*;

public class Main
{
    public static BufferedReader in;

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

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

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

iv Python 3

```python
num=int(input())

for i in range(num):
    name = input()
    print("Hello " + name + "!")
```

2 Solutions in Other Languages

i C#

```
using System;
namespace HelloCS
{
    class Program
    {
        static void Main(string[] args)
        {
            int nTests = Convert.ToInt32(Console.ReadLine());

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

NOTE If you develop your solution in Visual Studio, a number of libraries will be referenced with `using` statements. Many of these are not necessary (eg `System.Linq`) and may be removed.

ii Python 2

```
1 num = int(raw_input())
2
3 for i in range(num)
4     name = raw_input()
5     print "Hello " + name + "!"
```