DOMjudge Team Manual

(for South-Pacific Programming Contests 2013)

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 that can be found at
http://domserver.cosc.canterbury.ac.nz/domjudge/team.
See Figures 1 and 2 on the next page for an impression.

1 Submitting solutions

Solutions can be submitted from the web interface at http://domserver.cosc.canterbury.ac.nz/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 alphabet character and may contain only alphanumerical characters and {'.','_','-'}.

Note that the current compilers and versions on DOMjudge are Gnu cc/c++ 4.4, Oracle Java SDK 1.6, Python 2.6, Mono gmcs 1.6 (C# 2.0). 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, 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://domserver.cosc.canterbury.ac.nz/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. 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 '?'.

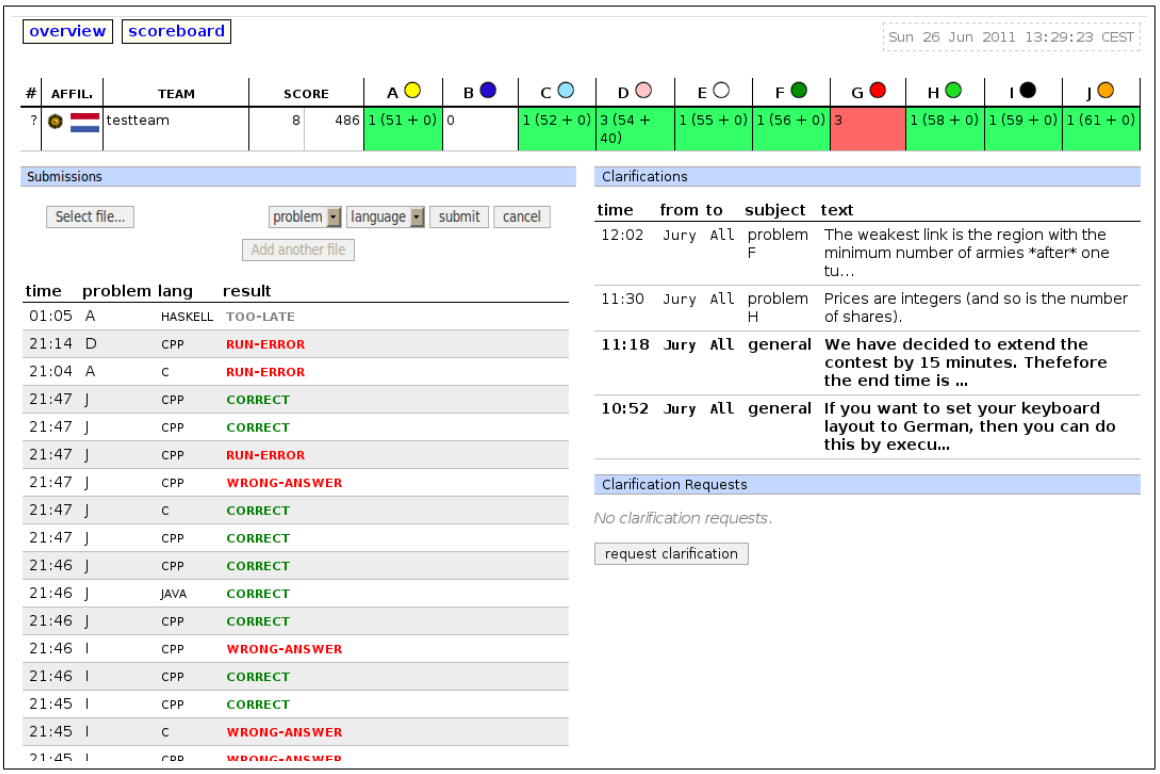

Figure 1: the team web interface overview page.

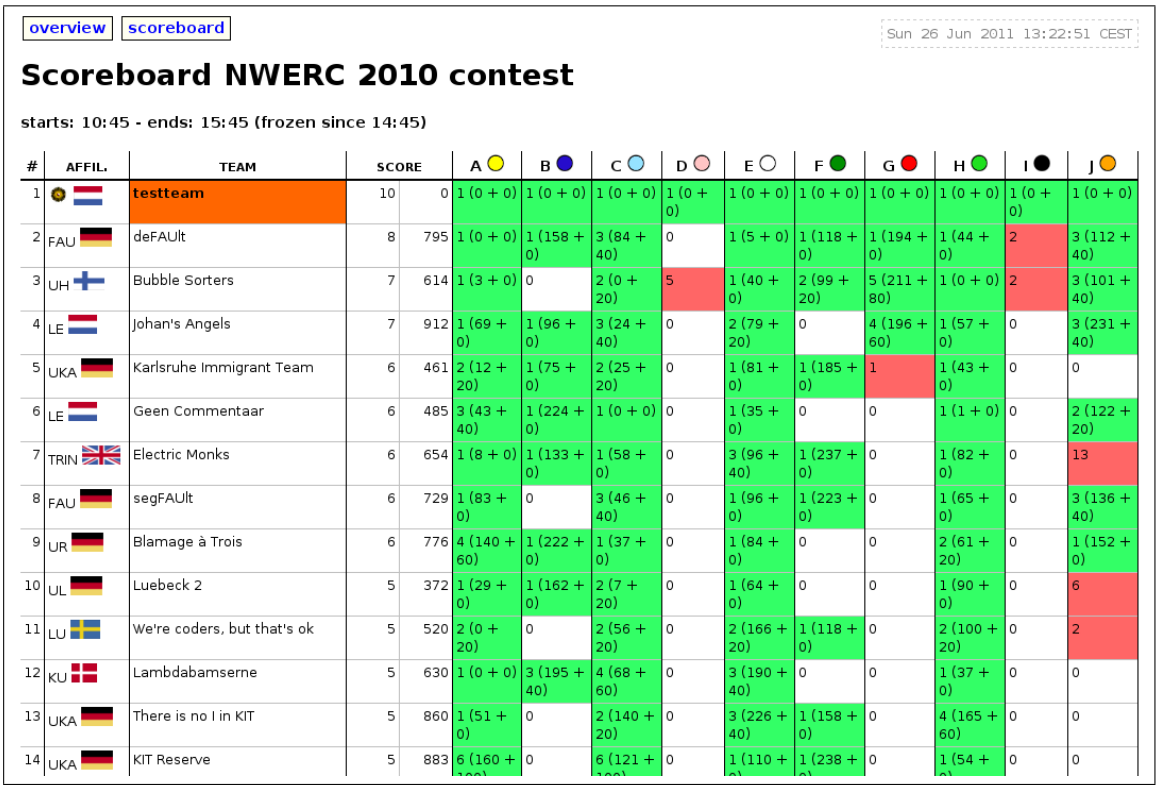

Figure 2: the scoreboard webpage.

Version/revision: 3.3.4 / 1f9e876
Last modified: January 20, 2013
Generated: August 12, 2013

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

Withthe 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).

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 2 gigabytes 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 (gcc 4.4):

```
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++ (g++ 4.4):

```
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 (Oracle Java 1.6):

```
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# (mono 2.6):

```
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 Python 2.6:

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