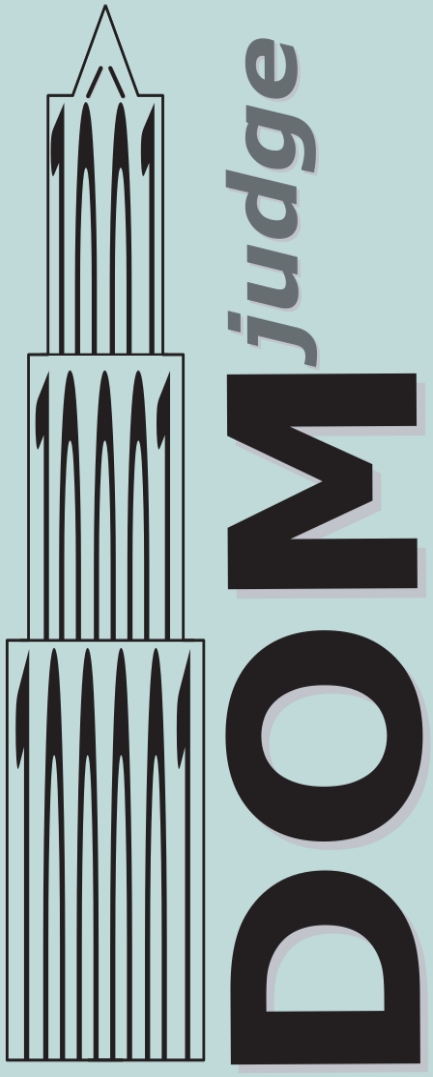

DOMJUDGE

Manual de equipo

Este manual proporciona una guía completa para interactuar con la interfaz web presentada a los equipos. Es aconsejable que todo el equipo lea todo el manual presentado.

Grupo de programación competitiva Universidad El Bosque - GPC UEB

Índice

Capítulo 1. Antes de comenzar

Previo al maratón de programación, su equipo debió hacer un pre registro en la organización promotora del evento, para que esta le provee un usuario y contraseña. Una vez con esta información, en la máquina donde fue asignado su equipo, debe iniciar un navegador web y digitar la dirección ip o la dirección web previamente otorgada.

Capítulo 2. Lectura y escritura

Este apartado especifica cómo se manejarán las entradas y las salidas. Las entradas de las soluciones deben ser leídas como "entrada estándar" y las salidas como "salidas estándar" (también conocida como consola) (Nunca se le pedirá interfaz gráfica para la ejecución de un problema). Usted nunca tendrá ni podrá abrir otros archivos desde el programa a ser ejecutado.

Capítulo 3. Interfaz web

Cabe mencionar que las imágenes mostradas en este documento, puede no ser iguales a las presentadas en la interfaz web una vez usted inicie sesión en su cuenta, debido a las actualizaciones que DOMJudge presenta.

3.1 Inicio de sesión

En este botón, usted podrá ingresar el nombre de usuario y la contraseña. (Vale recordar que, en la pantalla de inicio, en la parte superior derecha se puede observar el tiempo que falta para que el maratón inicie o termine según sea el caso).

3.2 Problemset

Se puede observar en la Figura 1. que esta ventana muestra los problemas a resolver en la competencia, la cual al seleccionar uno de ellos, se podrá ver el documento del ejercicio.

3.3 Scoreboard

La parte superior de la página muestra la fila de su equipo en el scoreboard: su posición y cuales problemas se intentaron resolver o se resolvió. A través del menú se puede ver el scoreboard público con las puntuaciones de todos los equipos. Muchas de las celdas mostraran información adicional.

La columna de puntuación lista el número de problemas resueltos y el total de tiempo incluyendo el tiempo de penalización. Cada celda de una columna de problemas lista el número de envíos, y si el problema fue solucionado, el tiempo del primer envío en minutos desde que inicio el concurso, aquí se incluye el tiempo total junto con cualquier tiempo de penalización incurrido por envíos incorrectos anteriores. Opcionalmente el scoreboard puede ser "congelado" por algún tiempo antes de finalizar el concurso.

Recuerde que el ganador será elegido por la mayor cantidad de ejercicios resueltos y si existiera un empate, se revisa quien es el equipo con el menor tiempo de penalización, lo que lo convertiría en el ganador del maratón.

Finalmente, a través del menú superior también se puede ver la lista de problema y ver/descargar textos de problemas y datos de muestra, si son proporcionados por los jueces.

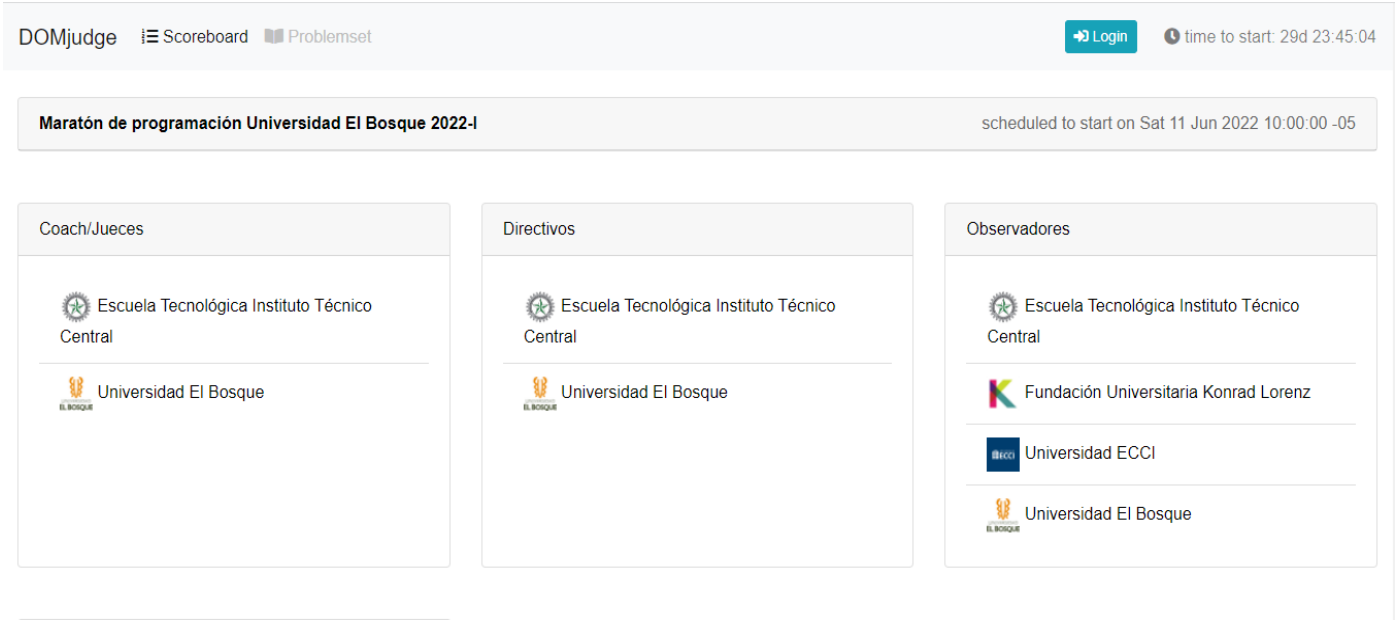

Figura 1. Interfaz inicial (DOMJudge) de equipo

Fuente: Sistema DOMjudge UEB

La Figura 2. Muestra cada uno de los posibles colores que vera en el scoreboard, las celdas podrán ser marcadas como:

3.3.1 Solved first

Es representado por el color verde oscuro, da a entender cuál fue el primer problema que fue desarrollado por un equipo.

3.3.2 Solved

Es representado por el color verde claro, muestra que problema ha sido solucionado.

3.3.3 Tried-Incorrect

Es representado por el color rojo, muestra que problema ha sido intentado, pero no ha sido posible darle solución.

3.3.4 Untried

Es representado por el color blanco, el cual por default siempre es mostrado al comienzo de una maratón. Da a conocer que problema no ha sido intentado.

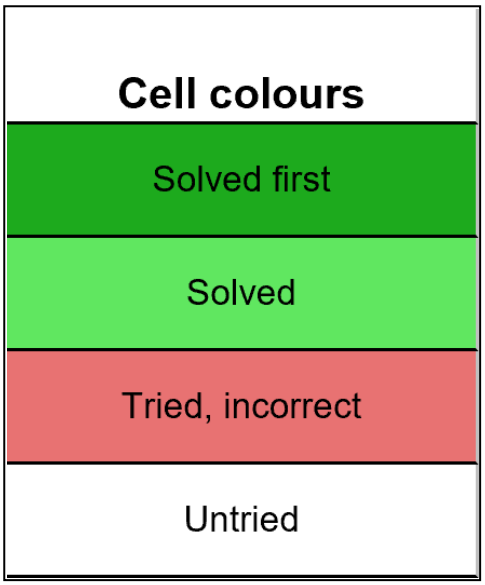

Figura 2. Distribución de color para la identificación de un problema

Fuente: DOMJudge Team Manual

Capítulo 4. Posibles resultados

La respuesta que le puede dar el juzgador puede tener múltiples resultados. Algunos de los descritos a continuación pueden o no aparecer (depende de la configuración del sistema).

Resultados	Descripción
CORRECT	Resolvió el problema. Cumple con todas las especificaciones.
COMPILER-ERROR	Error al compilar el programa. Note que cuando la compilación toma más de 30 segundos, será abortada y esto cuenta como error de compilación. El error de compilación no incurre en tiempo de penalización
TIMELIMIT	El código tomó más tiempo que el tiempo máximo permitido para el problema. Esto

	puede ocurrir porque se cuelga en un bucle o no es suficientemente eficiente.
RUN-ERROR	Error durante la ejecución de un programa. Esto puede ocurrir por tratar de dividir por cero, dirección incorrecta de memoria (por ejemplo, mediante la indexación de matrices fuera del límite), tratando de usar más memoria que el límite, etcétera. Verifique que su programa termina con el código de finalización 0.
NO-OUTPUT	El programa no genero ninguna salida. Compruebe que la salida es por consola.
OUTPUT-LIMIT	El programa generó más salida que el límite permitido. La salida se truncó y se consideró incorrecta.
WRONG-ANSWER	La salida de su programa es incorrecta. Esto puede suceder simplemente porque su solución no es correcta. Recuerde que debe cumplir exactamente con las especificaciones de los problemas. Preste atención en la cantidad de espacios impresos.
TOO-LATE	Cuando la operación de subir archivos sobrepaso el límite de tiempo de la competencia.

Tabla 1. Resultados de un problema

Fuente: DOMJudge Team Manual adaptado por Jhon Fredy Plazas Hurtado

Tenga en cuenta que los jueces preparan múltiples casos de prueba para cada problema, por lo tanto, las entradas que se pueden apreciar en un problema no son los únicos casos que el juzgador evaluará. DOMJudge reportara el primer resultado incorrecto como veredicto.

Capítulo 5. Tipos de usuarios

Dependiendo del usuario que este registrado en la maratón, se pueden diferentes tipos de categorías:

Figura 3. Tipos de usuarios

Fuente: DOMJudge Team Manual

Capítulo 6. Clarificaciones

Todas las comunicaciones con los jueces se deben hacer a través de clarificaciones. Esta puede ser encontrada en la columna derecha de la página de su equipo.

6.1 Clarifications

Aquí aparecen las aclaraciones que los jueces del evento realicen. Este puede ser enviado a todos los equipos de la competencia o tan solo uno. Los temas que se mostrará en este espacio pueden variar desde aclaraciones de un problema, hasta la respuesta a una pregunta que su equipo realizó.

6.2 Clarification requests

Aquí se podrá ver las preguntas que su equipo le ha emitido al juez.

6.3 Request clarification

Se abrirá una nueva ventana, en la cual puede solicitar aclaraciones de un problema del maratón (esto no quiere decir que se explicara cómo se desarrolla), los jueces pueden responder específicamente a su equipo o enviar una respuesta a todos si lo considera relevante. Recuerde que no se le dará de ningún modo la respuesta o se le ayudara para la compresión de un problema.

Capítulo 7. ¿Cómo son juzgados los envíos?

El sistema de DOMJudge es totalmente automatizado. El juzgamiento se termina al finalizar los siguientes pasos:

7.1 Enviado de soluciones

Con la interfaz web, usted puede subir la solución de un problema a los jueces. Recuerde que debe enviar el código fuente de su programa no el programa compilado o la salida (resultado) de su programa. Una vez enviado su código, entra en cola, esperando la compilación, ejecución y prueba, es por esto que dependiendo de la cantidad de envíos que el juzgador reciba, obtendrá los resultados de su envío en mayor o menor tiempo.

7.2 Compilación

Su programa será compilado en un juzgador automático que funciona en Linux. Todos los envíos serán pasados al compilador el cual genera un programa para su funcionamiento. Para Java y Kotlin la clase main será revisada; para Python 3 se realiza una revisión de sintaxis usando módulo de compilación py.

7.3 Pruebas

Después de que el programa sea compilado correctamente, será ejecutado y la salida que genere será comparada con la salida de los jueces. Antes de comparar la salida, el estatus de "salida" de su programa será verificado: si el código de salida de su programa no es cero, el resultado será RUNTIME-ERROR incluso si la salida de su programa es correcta. Recuerde que hay algunas restricciones durante la ejecución del programa, por lo cual, si es violada, será abortada con la respuesta RUNTIME-EROR.

Cuando se está comparando la salida de su programa, siempre se revisa que sea exactamente igual a la salida de los jueces, exceptuando algunos espacios en blanco, ya que estos serán ignorados (esto depende de la configuración del sistema). Llegado el caso un problema tenga diferentes soluciones (puntos flotantes o comas), el sistema modificará la función de comparación. Esto será aclarado en la descripción del problema.

Capítulo 8. Restricciones

Para mantener el sistema del juez estable y proporcionar a todos los competidores un entorno equitativo y justo, existen algunas restricciones por default (varían dependiendo de lo configurado por el administrador, se le avisará cualquier cambio si es necesario para el envió de un problema) las cuales serán descritas a continuación:

8.1 Tiempo de compilación

La compilación de su programa no debe tomar más de 30 segundos. Después de esto, la compilación será abortada y el resultado será COMPILATION-ERROR. En la práctica esto nunca deber dar problemas, por lo tanto, si llega a pasar en un programa normal, es mejor que se les informe a los jueces.

8.2 Tamaño de la fuente

El tamaño total del código fuente en un solo programa no debe exceder los 256 kilobytes, de otra forma él envió será rechazado.

8.3 Memoria

Durante la ejecución del programa, va a estar disponible 524288 kilobytes. Este es el tamaño total de memoria (incluye el código del programa, las variables dinámicas y estáticas, Java VM, entre otros). Si el programa que se envié trata de usar más memoria que lo permitido, será abortado, por lo cual resulta en un RUNTIME-ERROR.

8.4 Cantidad de procesos

No es necesario crear múltiples subprocesos (threads), ya que no servirá de nada, esto debido a que el programa que envíe, usará solo un procesador. Para incrementar la estabilidad del sistema, DOMJudge ejecuta los envíos en un SandBox en el cual se pueden realizar un máximo de 15 procesos.

Para las personas que nunca han programado haciendo uso de múltiples procesadores o que nunca hayan escuchado sobre "threads" no tienen por qué preocuparse ya que un programa normal trabaja en un procesador.

Capítulo 9. ¿Cómo enviar la solución de un problema?

9.1 Submissions

El recuadro más importante de la competencia, aquí es donde su equipo podrá enviar cada uno de los problemas resueltos. Para esto, una vez realizado el código de un ejercicio, guarde el archivo como lo indica la Tabla 2 luego seleccione el botón Browse y busque su código, seleccione el problema que va a enviar, el lenguaje que utilizó y envíe el ejercicio con el botón submit. Podrá observar que apareció una nueva fila con la hora de envió, el problema que escogió, el lenguaje y finalmente con el resultado del problema, las posibles salidas de éste, se puede apreciar en la Tabla 1.

Lenguaje	Nombre del archivo
JAVA	Main.java
PYTHON	Main.py
C++	Main.cpp
NOTA: En el caso específico de JAVA no se usa package, coloque su archivo fuente en el default package	

Tabla 2. Nombre del archivo para el envío de problemas

Fuente: Diego Fernando Rodríguez Castañeda

Capítulo 10. Tipos de problemas

Existen cuatro tipos de casos que pueden aparecer en los problemas:

Tipo de caso	Descripción	Ejemplo
Caso único	Sólo se utilizará una entrada por problema	N == 1
Caso bandera	El ejercicio seguirá ejecutándose hasta que una condición se cumpla	Mientras N sea diferente de 0
Caso fijo	Hay un número limitado de casos y el programa se ejecutará hasta que se cumpla este número	Mientras N sea menor a 10 Acumulando 1 por ciclo
Caso ilimitado o fin de archivo	El ejercicio seguirá leyendo y procesando entradas hasta que se encuentre el banderín EOF, se considera ese tipo como ilimitado.	Mientras siga habiendo entradas

Tabla 3. Tipos de problemas

Fuente: Diego Fernando Rodríguez Castañeda

Los posibles problemas que se destacan en un concurso, son los siguientes:

10.1 Ad Hoc

Es un tipo de problema donde cada descripción y su correspondiente solución son únicas. Estos pueden clasificarse en directos los cuales requieren solo la traducción del problema para luego ser codificado, o de simulación donde hay unas reglas que deben ser simuladas para obtener una respuesta.

10.2 Complete Search

Es un tipo de problema en donde se deben implementar búsquedas ya sean recursivas o iterativas.

10.3 Divide & Conquer

Consiste en dividir el problema en dos o más sub-problemas, más sencillos, de manera que la solución global sea la unión de las sub-soluciones. Algunos pasos para implementar este tipo de problemas son:

- Dividir el problema original en sub-problemas.
- Encontrar sub-soluciones para cada uno de los sub-problemas.
- Si es necesario, combinar las sub-soluciones para generar una solución completa al problema principal.

10.4 Greedy

Son los tipos de problemas relacionados con optimización. Consiste en elegir la opción más eficiente en cada paso local con la esperanza de llegar a la solución general más óptima. Pueden ser clásicos u originales.

10.5 Dynamic Programming

Es un procedimiento matemático diseñado principalmente para mejorar la eficiencia de cálculo de problemas de programación matemática seleccionados, descomponiéndolos en sub-problemas de menores tamaños y por consiguiente más fáciles de calcular.

10.6 Graph

Utilizados para representar mapas de rutas, organización de procesos, espacios de búsqueda para juegos, circuitos lógicos y demás. Un grafo es simplemente una conexión de vértices y enlaces que almacenan información conectada entre ellos, haciendo que su búsqueda sea más rápida.

10.7 Mathematics

Es la aplicación de técnicas matemáticas para la resolución de problemas algorítmicos. Puede existir una combinación entre algunos de los tipos de problemas ya mencionados.

10.8 String Processing

Es la solución de problemas mediante la utilización de estructura de datos, y algoritmos eficientes para cadenas.

10.9 Computacional Geometry

Son un conjunto de problemas con mucha más complejidad puesto que es más difícil encontrar el algoritmo correcto para dichas situaciones. Pueden estar relacionados con geometría básica o geometría computacional.

10.10 Some Harder Problems

En este caso puede haber una combinación de varios tipos de problemas, o que estos no entren en ninguna de las categorías mencionadas con anterioridad.

Capítulo 11. Ejemplos de códigos

Los siguientes ejemplos presentados a continuación muestran como es la entrada y la salida para cada tipo de lenguaje disponibles en la plataforma (depende de la configuración del administrador).

Los ejemplos son soluciones para el siguiente problema Figura 4: la primera línea de entrada contiene el número de casos. Luego cada caso consiste en una línea, la cual contiene un nombre (una sola palabra) con tamaño máximo de 90 caracteres. ¡Para cada caso de prueba, se debe imprimir “Hello <nombre>!” en una línea separada.

La entrada y salida para este problema es (note que el número 3 indica la cantidad de casos):

Input	Output
3 world Jan SantaClaus	Hello world! Hello Jan! Hello SantaClaus!

Figura 4. Ejemplo de entrada y salida de un problema

Fuente: DOMJudge Team Manual

11.1 Solución en 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);
        }
}
```

Figura 5. Ejemplo de código en C

Fuente: DOMJudge Team Manual

11.2 Solución en 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;
        }
}
```

Figura 6. Ejemplo de código en C++

Fuente: DOMJudge Team Manual

11.3 Solución en Java

```java
import java.util.*;

class Main {
        public static void main(String[] args) {
                Scanner scanner = new Scanner(System.in);
                int nTests = scanner.nextInt();

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

Figura 7. Ejemplo de código en Java

Fuente: DOMJudge Team Manual

11.4 Solución en Python

```python
import sys

n = int(input())
for i in range(n):
    name = sys.stdin.readline().rstrip('\n')
    print('Hello %s!' % (name))
```

Figura 8. Ejemplo de código en Python

Fuente: DOMJudge Team Manual

11.5 Solución en 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+"!");
                }
        }
}
```

Figura 9. Ejemplo de código en C#

Fuente: DOMJudge Team Manual

11.6 Solución en 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.
```

Figura 10. Ejemplo de código en Pascal

Fuente: DOMJudge Team Manual

11.7 Solución en Haskell

```haskell
import Prelude

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

Figura 11. Ejemplo de código en Haskell

Fuente: DOMJudge Team Manu

LICENCIA

Se otorga la licencia GNU Lesser General Public License (LGPL) al presente manual de usuario.

Se respeta y se mantienen las licencias GNU General Public License versión 2 (GPLv2), BSD license y MIT/X11 license, con que la comunidad de desarrollo de DOMJudge lo libera.