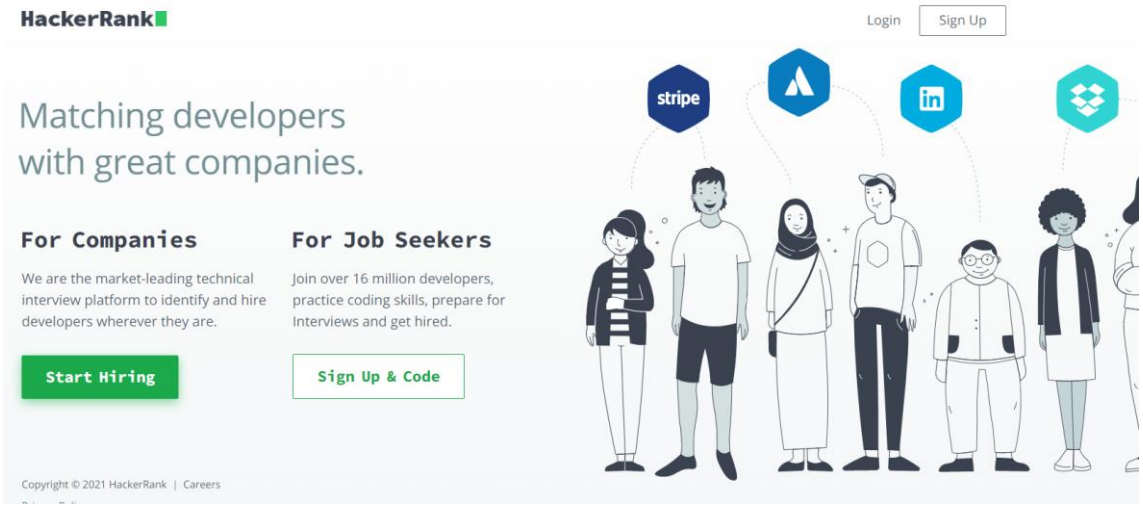
Descripción de la plataforma

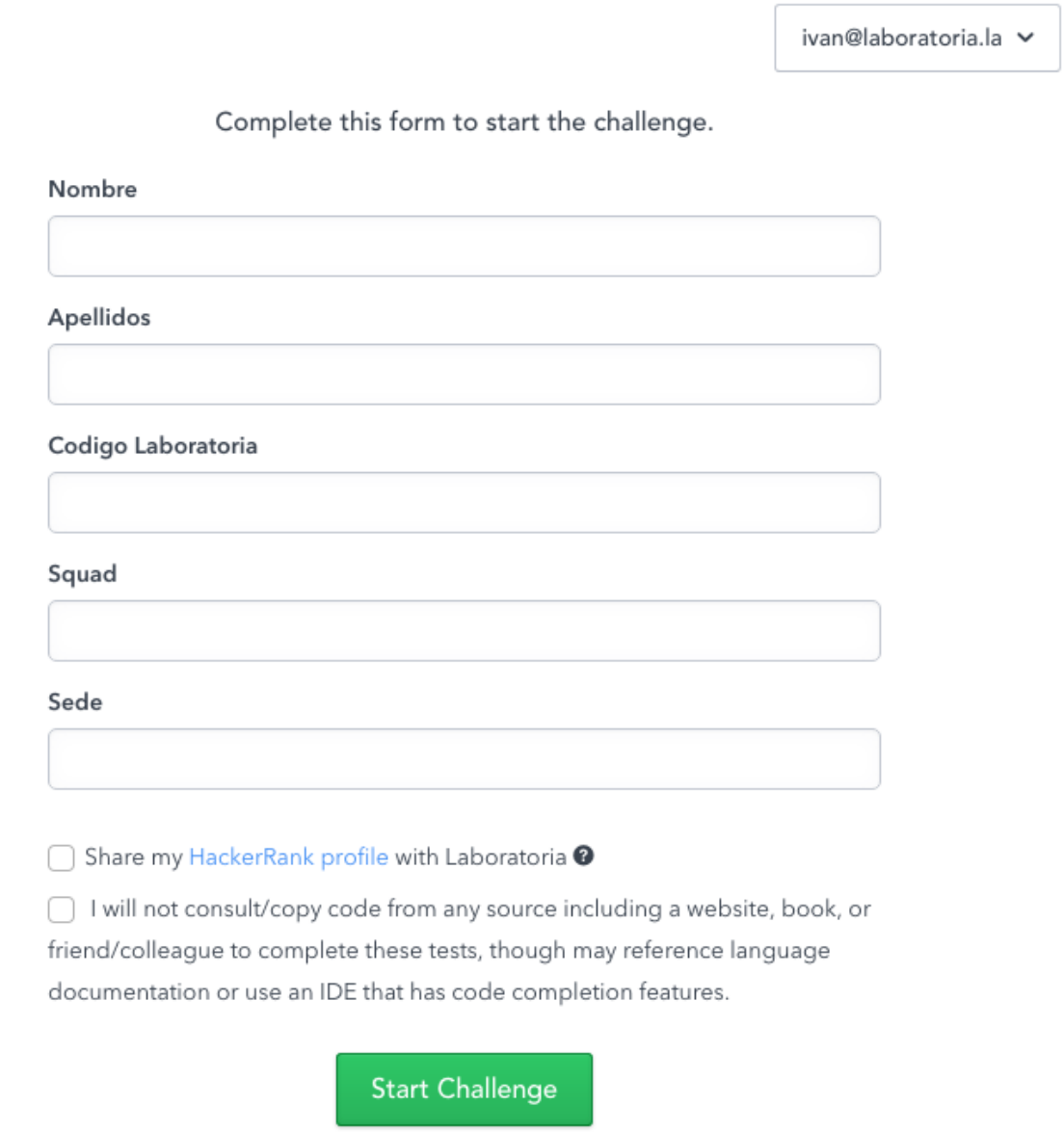
Introducción

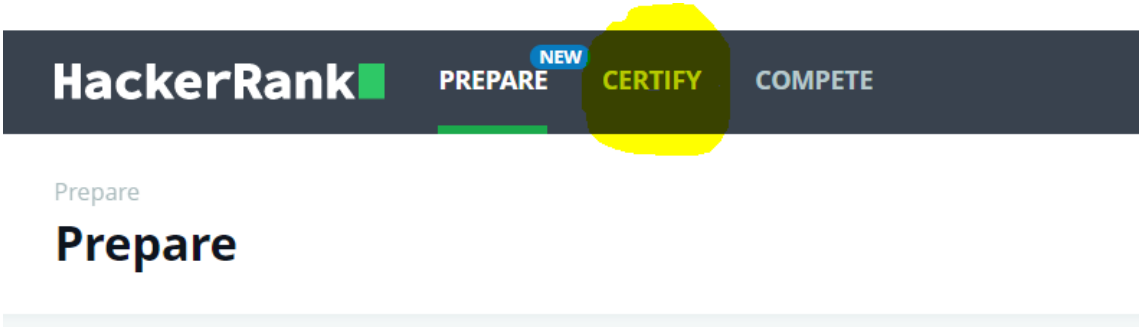
Hackerrank es una plataforma online pensada para realizar competiciones de programacion.

Funcionamiento interno

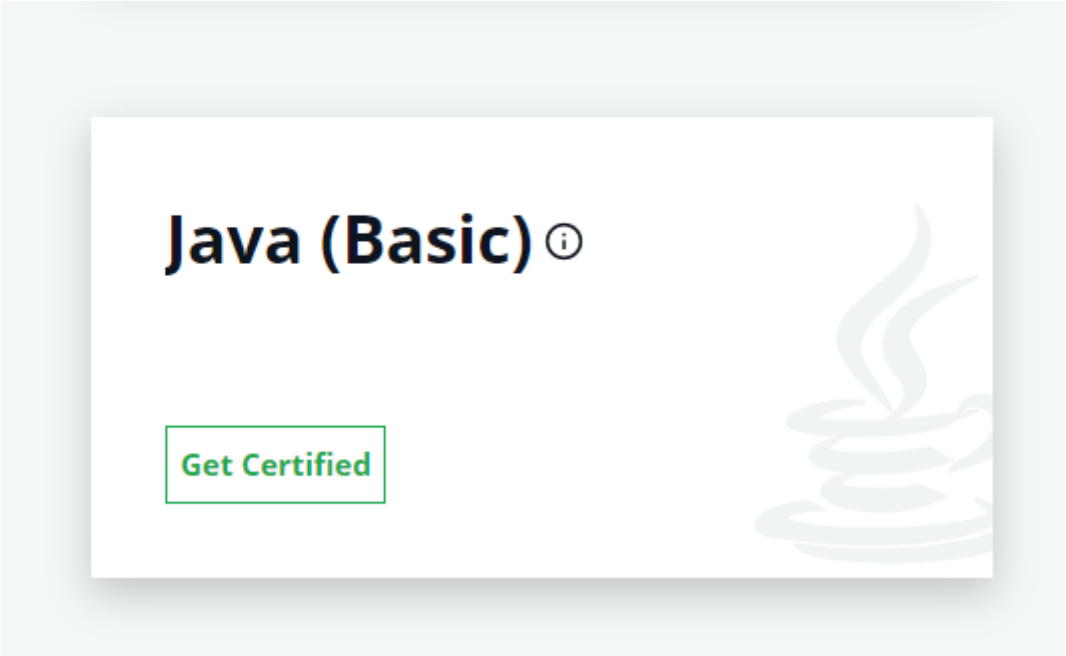


- la primera vez que entremos a HackerRank, debemos de crear una cuenta con solo ingresar 3 datos (*nombre completo*, *correo electrónico*, *contraseña*) y dar clic en **Sign up**.
- Una vez registrado, nos mostrará un formulario (tal como se ve en la imagen debajo) que deberemos completar y marcar ambos checks (requisito de HackerRank) para ingresar al test.

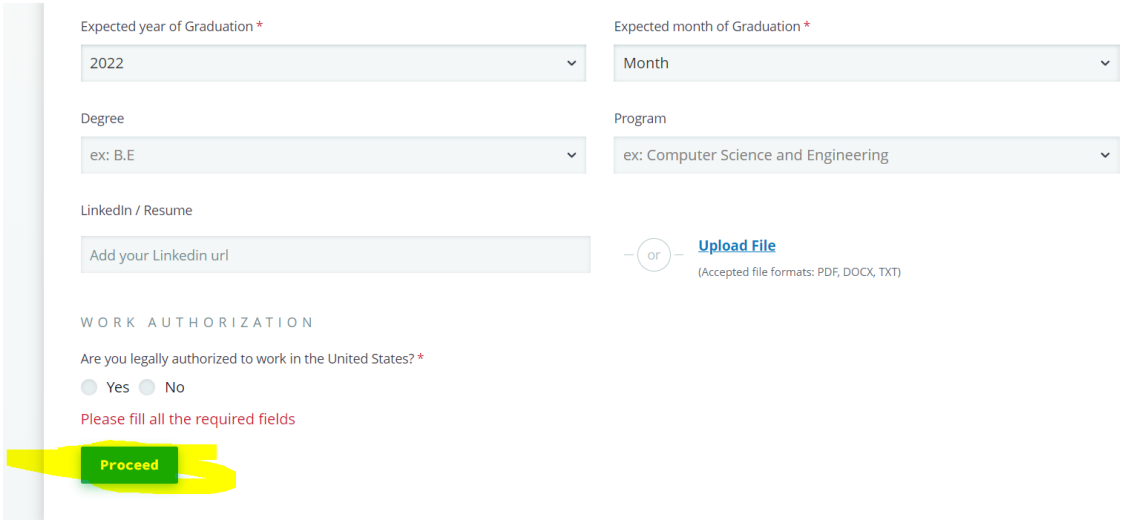


- 

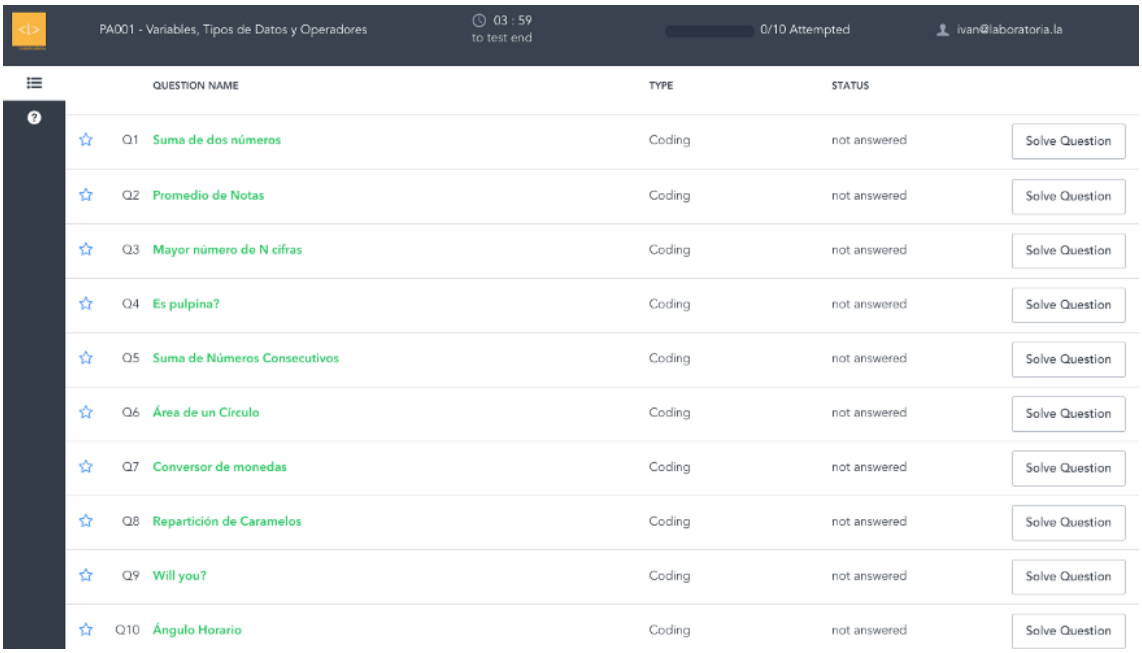
Y dentro de las certificaciones buscamos la correspondiente, por ejemplo Java Basic:



Y procedemos a cumplimentar nuestros datos:

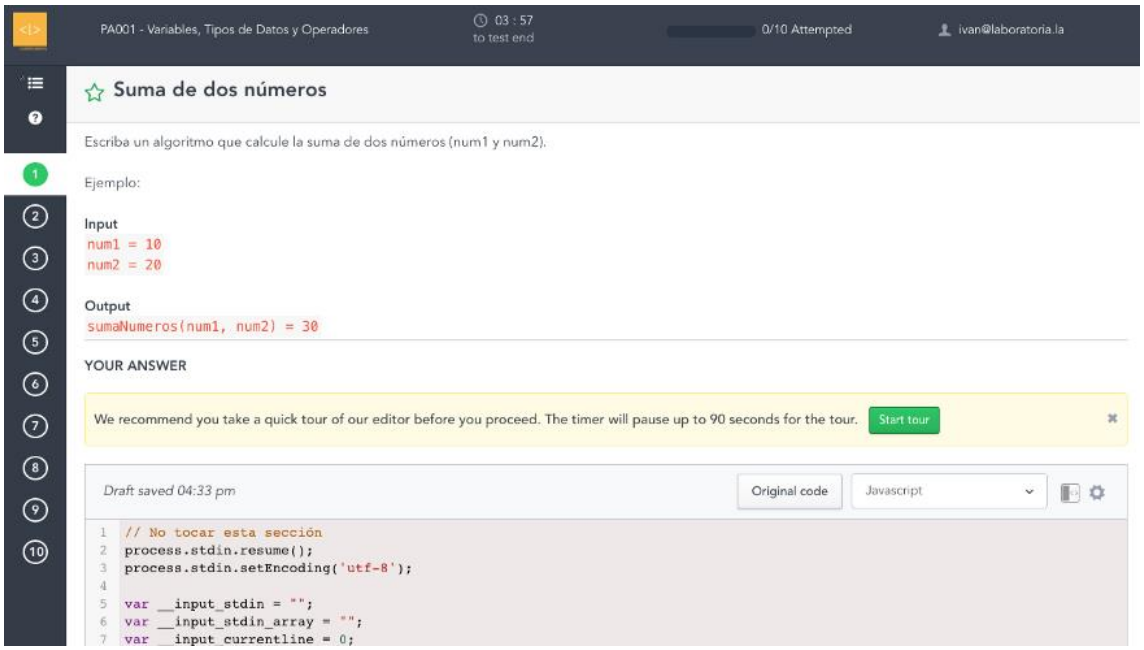


- Una vez dentro del test, veremos el tiempo restante para culminarlo así como las preguntas del test:



Test de HackerRank

- Para resolver un test, solo es necesario dar clic al nombre del ejercicio y nos llevará a una pantalla con el enunciado de la pregunta y un editor de código embebido:



Ejercicio de un test de HackerRank

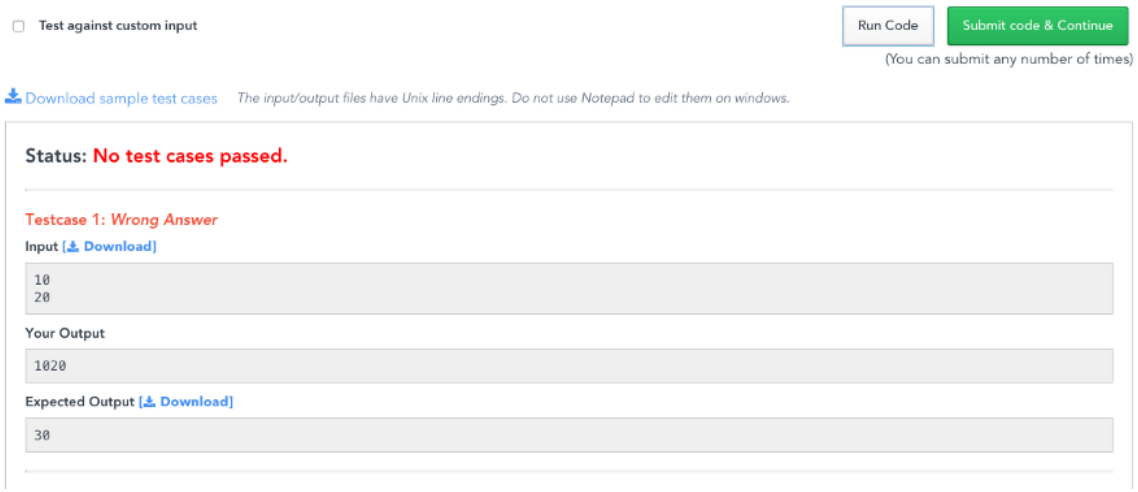
- Dentro del código que viene por defecto en el editor, se encuentra una función, esta es la que se ejecutará al momento de enviar tu solución para comprobar que los parámetros (**entrada**) mediante la función (**algoritmo**) obtengan el resultado correcto (**salida**).

```
/*
 * Implementar el algoritmo dentro de la función.
 */
function sumaNumeros(num1, num2) {
    var suma = num1 + num2;
    return suma;
}
```

num1 y num2 son las entradas, en este caso, suma es la salida.

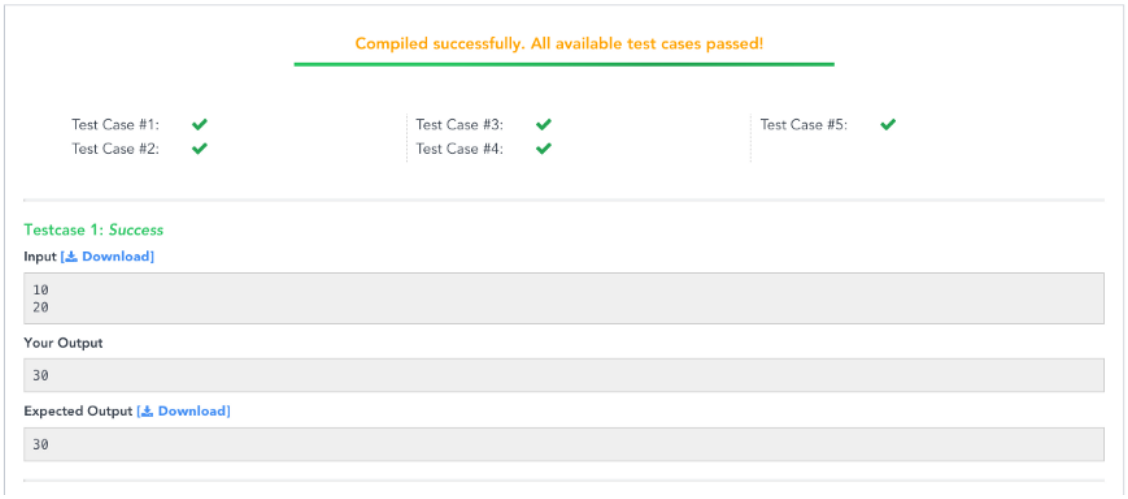
Nota: *Debe de escribirse la palabra clave* ***return*** *para que la función retorne el resultado que se espera.*

- Luego de implementar la solución pensada, aparecen 2 botones **Run Code** y **Submit code & Continue**. El primer botón, permite ejecutar la función reemplazando los parámetros por casos de prueba y comparar el resultado esperado con el resultado obtenido. El segundo botón, hace lo mismo que *Run Code* pero envía tu solución y cierra el ejercicio entendiendo de que ya no volverás a editar tu solución.



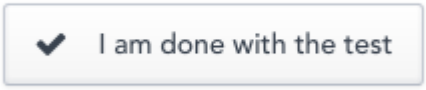
Run Code.

- Si te sale el mensaje en rojo **Wrong Answer**, quiere decir que la solución no cumplió los casos de prueba. La sección debajo de ***Input*** son los valores que reemplazarán a los parámetros de la función. ***Output*** es el resultado que devuelve la solución (lo que pusiste luego de return en el código). ***Expected Output*** es el resultado esperado. Para que tu solución esté correcta, el *Output* y *Expected Output* deben ser exactamente lo mismo.



Casos de prueba correctos

- Cuando todos los casos de prueba pasan correctamente, debes enviar tu solución dando clic en *Submit code & Continue*. HackerRank te redireccionará al siguiente ejercicio automáticamente.
- Cuando quieras terminar el test debes volver a la pantalla principal del listado de ejercicios y darle clic a **I am done with the test**.



Cada problema tiene asociada una puntuacion maxima que sera conseguida cuando el codigo que suba el usuario pase todos los casos de entrada a los que se le somete.
Supongamos que existen 10 casos de entrada en un determinado problema; si el codigo subido por el usuario supera 6 de los 10 casos de entrada, la puntuacion obtenida en ese problema sera de 60 puntos (6/10 * 100 puntos).

Un problema puede ser intentado cuantas veces se quiera. Para esta competicion no habra penalizacion por intentos fallidos.

Dependiendo del lenguaje empleado existira un tiempo maximo de ejecucion del problema. Algunos casos de entrada se han escogido para forzar este tiempo maximo, por lo tanto tu codigo debe estar optimizado. Mas informacion sobre el tiempo maximo de ejecucion y librerias permitidas
en: https://www.hackerrank.com/environment

Por favor, si vais a usar Java / C# / Clojure / Scala debeis llamar a la clase subida "Solution" (mas info. en el link de arriba).

Descripción de los problemas

Cada problema tendra una descripcion suficientemente detallada de la tarea a programar por el participante. Junto a la descripcion se proporcionar el formato de los datos de entrada que el programa del participante debe recibir. Los datos **SIEMPRE** seran leidos de la entrada estandar (STDIN) y los resultados deberan ser **SIEMPRE** mostrados por la salida estandar (STDOUT).

La manera en que deben leerse los datos de entrada y mostrarse los datos de salida depende del lenguaje usado, pero aqui hay algunos ejemplos:

C:
IN: scanf() gets() getchar()
OUT: printf() puts() putchar()

C++:
IN: cin
OUT: cout
C#:
IN: Console.ReadLine()
OUT: Console.WriteLine()
Python2:
IN: raw_input()
OUT: print "..."
Python3:
IN: input()
OUT: print()
Java:
Ver http://en.wikipedia.org/wiki/Standard_streams#1995:_Java
Entrada

El formato de los datos de entrada sera proporcionado en la seccion *Input Format* (debajo de la descripcion). Le sigue inmediatamente las "Constraints" o restrcciones para la entrada. Indican el tamaño maximo de los datos de entrada y el rango para los numeros
Salida

El formatode los datos de salida sera proporcionado en la seccion *Output Format* (debajo de *Input Format*)
Ejemplos de entrada y salida

Junto a los formatos de entrada y salida se proporcionara un ejemplo para que el participante entienda mejor el problema. El ejemplo **SIEMPRE** formara parte de los casos de entrada contra los que se evalua el problema.

Problema de ejemplo

Para que entendais todos el funcionamiento de la plataforma vamos a suponer un caso sencillo.

El problema contendria el siguiente enunciado:

Se quiere instalar un sencillo Siri en la puerta del edificio de Teleco. Cada vez que una persona entre al edificio esta dira su nombre y el Siri debe darle los buenos dias. Asumimos que existe la tecnologia de Voz a Texto (Speech Recognition), de tal manera que tu programa tan solo recibira el nombre de la persona y debe saludarla correctamente.

(Abajo sigue el formato de entrada y salida)

Input Format

Una sola linea (acabada en salto de linea) con el nombre de la persona a saludar.

Constraints

Longitud del nombre de la persona < 100 caracteres

Output Format

La frase "Hola XXXX" donde XXXX es el nombre de la persona leido de la entrada estandar. Acabara siempre con un salto de linea.

Sample Input

Perico el de los palotes

Sample Output

Hola Perico el de los palotes

Explanation

"Perico el de los palotes" es leido de la entrada estandar (por ejemplo raw_input() con python 2) y la frase "Hola Perico el de los palotes" es mostrada (por ejemplo con print).

Ese programa en Python2 se acometeria de la siguiente manera:

```
nombre = raw_input()
print "Hola " + nombre
```

Al final, si hemos conseguido superarlo, nos debe aparecer una certificación de este tipo.

Java (Basic) Certificate

Google Translate

Share this Certificate

https://www.hackerrank.com/cert... Copy Link

Java (Basic)

It will cover basic topics in Java language such as classes, data structures, inheritance, exception handling, etc. You are expected to be proficient in either Java 7 or Java 8.

Standout from the crowd

Take the HackerRank Skills Certification Test and make your profile stand out to employers and peers

Learn More