DOMjudge 說明文件

October 30, 2023

DOMjudge 參賽者使用說明

October 30, 2023

Contents

1 教學

1.1 開始使用 DOMjudge

進入 DOMjudge 頁面後，點選右上方的「Login」來登入 DOMjudge，如果運作正常的話，你會看到類似於以下的畫面：

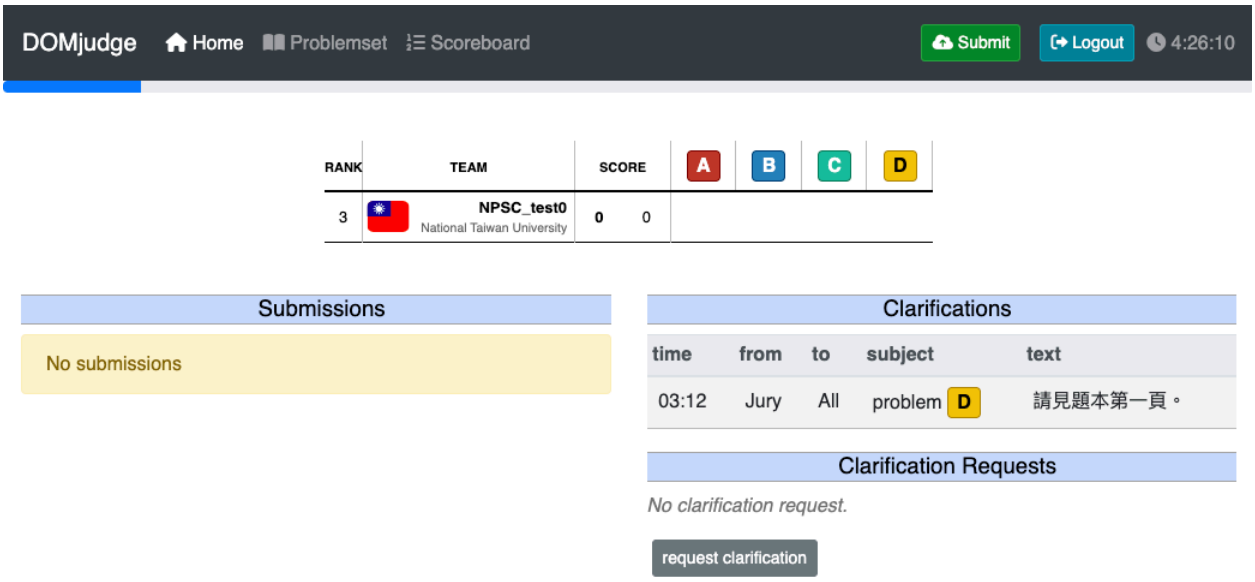

Figure 1: 登入畫面

在最上方的選項中

- Home 代表著你的隊伍介面，也就是上圖中顯示的畫面，關於該介面的詳細說明可以去隊伍介面觀看。
- Problemset 代表著題目列表，點選該頁面可以看到當前比賽的題目列表，並可以在裡面獲得題目敘述、範例測試資料、題目限制等資訊。
- Scoreboard 代表著計分板，點選該頁面可以看到當前比賽所有隊伍的解題狀況以及排名。
- 右上角的時間代表著比賽剩餘時間，必須把握時間解出題目！

1.2 撰寫第一道題目

接著如果試著寫一題題目就再好不過了。最簡單的題目應該是北極熊大遷徙，這個問題要求你上傳一個會輸入兩個數字後、輸出兩個數字總和的程式。

用 C, C++ 這些語言寫個能解決這個問題的程式（可以參考程式碼範例），接著就可以參考如何上傳程式碼來上傳這個程式了！

2 如何上傳程式碼

NPSC 目前提供上傳程式碼到 DOMjudge 的方法只有透過網頁介面，以下是上傳方法的介紹：

2.1 透過網頁介面

要上傳解答，你需要透過 DOMjudge 提供的網頁介面進行上傳。只要處在已登入的狀態，你都可以在每一個頁面中找到上傳用的綠色「Submit」按鈕，點擊後就可以再點擊跳出視窗裡「Source files」的「Browse」來選擇檔案。**請務必檢查你上傳的檔案是「程式碼」，而不是編譯後的「執行檔」**。

選擇檔案後，DOMjudge 會率先嘗試利用檔名和副檔名來自動辨識欲上傳的題目（欄位「Problem」）以及要使用的語言（欄位「Language」）。若偵測錯誤或失敗，則你會需要再自行指定下方的欄位。請注意，你上傳的檔案名稱必須以字母或是數字開頭、並且整個檔案名稱只能包含字母、數字以及額外的四種字元 +._-。

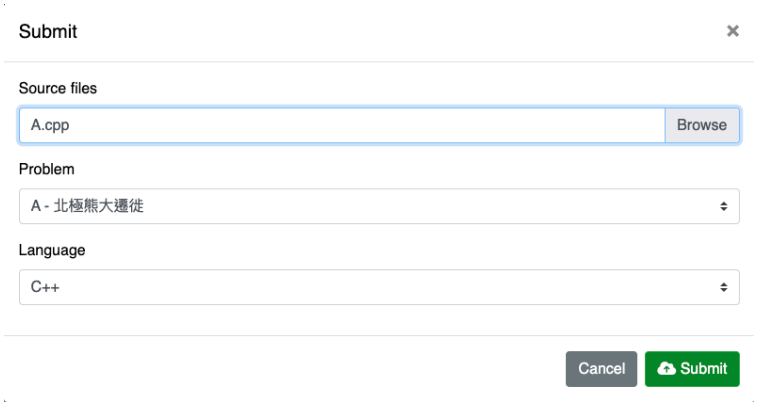

Figure 2: 上傳 A.cpp 時 DOMjudge 會自動認為你打算用 C++ 上傳問題 A

確定資訊沒問題後，點擊跳出視窗裡的「Submit」並確認後，DOMjudge 就會先將頁面導向你的隊伍介面。

3 隊伍介面

點選 DOMjudge 選單的「 Home 」，就可以看到所有與自己隊伍相關的資訊。

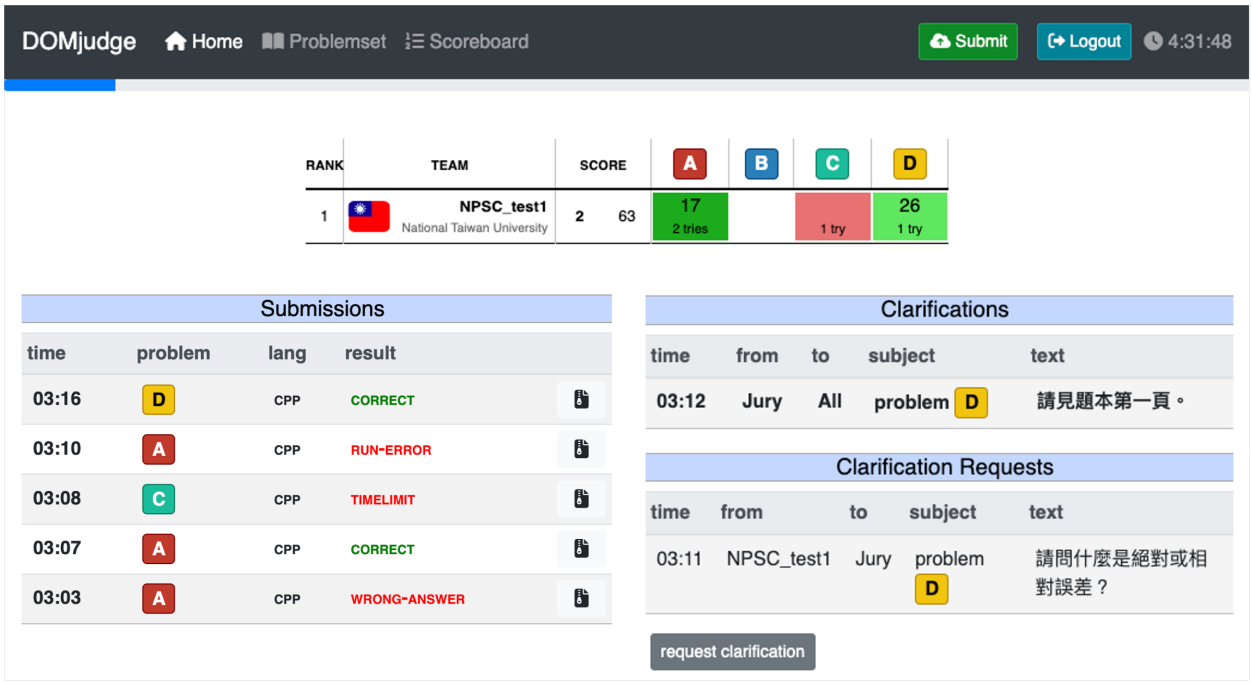

Figure 3: 隊伍介面範例

隊伍介面主要分成左方的「 Submission List 」、上方的「 Team Row 」以及右方的「 Clarifications 」三個部分，下面我們會一一介紹。

3.1 Submission List

Home 的左邊顯示的是你所有 submission 的資訊，包含上傳時間、上傳的語言、上傳的題目以及裁判結果。對於每一筆 submission，點開則可以看見編譯時的訊息，點選右方的按鈕則可以下載當時上傳的程式碼。有關於可能的裁判結果可以參考可能的裁判結果。

3.2 Team Row

Home 的上方顯示的是你的隊伍現在的所在名次和解題狀況，左方的分數欄會顯示已通過的題目數量以及罰時的總和，而題目列的每個格子也會列出你的隊伍上傳該題的數量，如果你的隊伍已經通過了該題，則會再額外顯示你自從比賽開始後第一次通過的時間（以分鐘為單位）。

你可以透過選單裡的 Scoreboard 來看到所有隊伍的解題狀況，但計分板有可能會在比賽結束前的某個時間被「凍結」，此時若檢視 Scoreboard，所有隊伍在「凍結」後的上傳

都會用藍色來表示有未知的傳送結果，但隊伍自己的 Home 則會正常顯示自己隊伍上傳狀況，唯隊伍排名會被更改為「?」。

3.3 Clarifications

Home 的右邊顯示的是所有與裁判相關的互動結果，無論是針對比賽的公告、還是隊伍針對題目給裁判的詢問與其回覆都會列在這裡。

所有跟題目相關的問題都必須透過 Clarifications 系統與裁判溝通，若你需要進行提問，可以點選下方的「request clarification」來撰寫一個提問。提問時，你會需要先在「Subject」選擇一道題目做為詢問的主題，若不是特定題目的問題，則可以選擇「General issue」來當作主題。接著，在底下的 Message 明確的寫出你的疑問後，按下「Send」就可以送出，裁判會在收到你的提問後儘速回覆。

注意！在比賽過程中請務必定時留意該區域，無論是公告還是裁判給你的回覆，都有可能會影響比賽時對題目的理解，甚至可能會影響到比賽狀況，千萬別漏掉重要訊息。

3.4 可能的裁判結果

當你上傳一筆 submission 後，DOMjudge 會先將你的 submission 在 Home 的左方顯示為「PENDING」，待評測完畢後，相對應的裁判結果就會被顯示出來。以下是所有可能的裁判結果。

裁判結果	說明	是否罰時
CORRECT	指 DOMjudge 對你的程式感到滿意，你正確地解開了這題，恭喜你！	No
COMPILER-ERROR	指 DOMjudge 無法編譯你的程式，為了幫助你找到問題，你可以點開該 submission 來看到編譯器輸出的額外訊息。比賽環境的編譯器與參數設定可以在 Technical information 頁面找到。	No
RUN-ERROR	指你的程式在執行過程中，因為我們的測試資料而造成非預期的結束。更精確地說，可能的原因有： 1. 程式結束時的回傳值非零 (non-zero exit)，或是有 exception 發生。 2. 你的程式嘗試使用了超過題目規定的記憶體上限。 注意因為使用程式結束的回傳值 (exit code) 來決定你的程式是不是正常的結束，所以請確定你的 main 函式不要回傳非零的值。	Yes
TIMELIMIT	指你的程式執行的時間過長，最後超過執行時間限制時被終止。請注意，所有輸出的結果在程式正確結束之前都不會被檢查，所以拿到 TIMELIMIT 不代表你的程式目前產生的結果是正確的。	Yes
WRONG-ANSWER	指你的程式在規定的時間內結束，但是你的答案不正確。因為沒有更多的資訊，所以這通常是個最令人沮喪的結果。有時候唯一的方法就是自己生些測試資料來嘗試找到 bug。	Yes
OUTPUT-LIMIT	指你的程式產生過多的輸出，以致你的程式被強迫終止，以確保輸出不會佔滿我們的硬碟。請確定你沒有困在無窮迴圈內並且一直輸出東西，或是有正確地處理輸入。	Yes
NO-OUTPUT	指你程式沒有任何輸出，最好檢查一下你的程式是不是忘了輸出東西！	Yes
TOO-LATE	指程式在比賽結束後才上傳，已經太遲了，下次再努力。	No

注意到題目可能會擁有多筆測試資料，若你上傳的 submission 有錯誤，DOMjudge 會顯示該 submission 第一個錯誤的測試資料所產生的裁判結果。

4 Technical information

4.1 輸入與輸出

你撰寫的程式要從**標準輸入（standard input）**來讀取（file descriptor 0, `STDIN_FILENO`/`stdin` in C, `cin` in C++），並從**標準輸出（standard output）**來輸出（file descriptor 1, `STDOUT_FILENO`/`stdout` in C, `cout` in C++）。任何輸出到**標準錯誤輸出（standard error）**的訊息都會被忽略（file descriptor 2, `STDERR_FILENO`/`stderr` in C, `cerr` in C++）。如果你好好利用這個特性，可以讓你在除錯時更方便。（舉例來說，你如果將你的所有的除錯訊息都輸出至標準錯誤輸出，則在上傳的時候可以不用將這些除錯訊息刪除。）但要注意的是，這些額外的輸出仍會消耗你的執行時間。

所有的輸入都會符合題目中的 **Input** 說明。如果題目中說輸入是個正整數，你就不需要處理輸入是負整數的情況（也不需要處理不是整數的輸入）。切記，要看清楚輸入的格式，有時候輸入是很邪惡的！輸出時 DOMjudge 會要求輸出格式與題目敘述中的 **Output** 相符，儘管一些多餘的空白並不影響評分結果，但任何可見的差異都會被評測為錯誤！

4.2 編譯與編譯器

你上傳的程式會被自動放在一台 Linux 的機器裡被編譯器編譯成一個單一執行檔並執行，以下是編譯你的 submission 時的編譯器跟編譯器參數

Language	編譯器	版本	參數
C	GCC	gcc version 10.2.1 20210110 (Debian 10.2.1-6)	`-O2 -std=c11 -static -lm`
C++	GCC	g++ version 10.2.1 20210110 (Debian 10.2.1-6)	`-O2 -std=c++17 -static`

舉例來說，如果你上傳 `main.cpp` 並使用 C++ 編譯，編譯過程會執行 `g++ -O2 -std=c++17 -static main.cpp`。

4.3 系統函式庫

你可以任意地用你撰寫語言的標準函式庫與類別（class）。像是 C++ 有 STL (Standard Template Library)。STL 的版本會跟 GCC 編譯器的版本相同。

4.4 程式評測過程

當你上傳的程式碼成功編譯成功後，DOMjudge 會執行你的程式並予以裁判預先設置好的測試資料輸入進你的程式裡、並將程式的輸出儲存起來與裁判的輸出作比對，此時一些額外的狀況可能會發生：

- 程式超過執行時間限制：如果你的程式花費超過該題的執行時間限制（可以在 Problemset 的頁面找到每題的時間限制），你的程式會被判斷為 TIMELIMIT 並直接結束評測。
- 程式非預期的結束：當程式結束時，DOMjudge 會取得你程式的回傳值（exit code）。如果你的回傳值非 0，你的程式會被判斷為 RUN-ERROR。

若額外狀況沒有發生，DOMjudge 就會正常的比對兩個輸出檔並回傳結果，在一般的狀況下，你的輸出必須要跟裁判的輸出在去除多餘的空格後一模一樣才會被視為 CORRECT，但若題目的答案可能會有很多種（例如浮點數的輸出），則裁判會設置好特殊的比對器來進行比對，該比對方式會撰寫在題目敘述裡面。

請注意，裁判可能會設置多組測試資料，對於每一筆測試資料 DOMjudge 都會用相同的流程執行過一遍，只要 DOMjudge 先行偵測到一筆測試資料有錯誤，整個評測過程就會被中止並將該筆測試資料的裁判結果回傳給你。DOMjudge 測試每個測試資料的時候，都會開一個全新的執行緒來執行你的程式，也就是你可以假設每一筆測試資料間的測試是獨立的。

4.5 額外限制

為了防止惡意行為、保持系統的穩定性並給予參賽選手乾淨且相同的執行環境，每一個 submission 都有一些共同的限制：

編譯時間	你的程式不可以花費超過 30 秒的編譯時間，否則 DOMjudge 會直接中斷編譯的過程並給予你 COMPILER-ERROR 的裁判結果。實務上這不應該會造成比賽的爭議性，若你在正常的情況下遇到了此問題，請立刻聯絡裁判。
程式碼大小	你上傳的程式碼檔案大小不可以超過 256KB，若超過此大小上限，DOMjudge 會直接拒絕你上傳該份程式碼。
記憶體限制	在 Problemset 的頁面裡，你可以找到每題的記憶體限制，代表著該題可以使用的記憶體大小上限，包含程式碼本身、靜態或動態宣告的變數、stack、……等等。如果你的程式使用了超過該題上限的記憶體，DOMjudge 會中止你的程式並給予你 RUN-ERROR 的裁判結果。
平行運算	理論上我們不預期有人會使用平行化的程式技巧，意即創造多個「執行緒」，這是因爲在 DOMjudge 執行程式碼的過程中，你的程式只會恰被一個執行緒執行，所以這實際上是沒有用的！

5 比賽規則

5.1 名次比較方式

在比賽中，你的名次主要決定於你解了多少題（scoreboard 中 SCORE 那欄左邊的數字），如果平手的話我們會再看你解題的總時間，又稱總罰時（scoreboard 中 SCORE 那欄右邊的數字），如果還是平手的話，會比較最後通過那題的上傳時間。

5.2 解題數

當你在比賽結束前成功上傳，並獲得 CORRECT 的裁判結果，即表示你解出了該題。解題數意即你獲得 CORRECT 的題目數量，解題數多的獲勝。

5.3 總罰時

如果兩個參賽者解決了同樣數量的題目，總罰時比較少的人名次會比較前面，總罰時的定義是所有題目初次獲得 CORRECT 的 submission 上傳時間加上因為錯誤 submission 而造成的罰時。只有已經解出的題目會加到總時間的計算中。在該題成功解出前的每個錯誤的 submission 會造成 20 分鐘的罰時。所謂的錯誤的 submission 是指 RUN-ERROR、TIMELIMIT、WRONG-ANSWER、OUTPUT-LIMIT 和 NO-OUTPUT 這五種 judge 結果。（請見可能的裁判結果）。

總罰時計算例子

如果說有個參賽者有

- 兩個錯誤的 submission 在 Problem A，並且在 30 分鐘時解決題目
- 三個錯誤的 submission 在 Problem B，但沒有完成題目
- 一個錯誤的 submission 在 Problem C，並且在 45 分鐘時解決題目

這參賽者解決了兩題題目，因為 Problem B 最後沒有被解決，所以 Problem B 的錯誤並不會加到總時數的計算中。Problem A 貢獻 $30 + 2 \times 20$ 分鐘，Problem C 貢獻 $45 + 1 \times 20$ 分鐘，所以最後總共是 135 分鐘。注意到由於我們都是用分鐘做為單位，因此無論你是在第 30 分鐘 00 秒上傳、還是在第 30 分鐘 59 秒上傳，都會被視為是在第 30 分鐘上傳的。

5.4 最後通過時間

如果兩個參賽者解了同樣的問題數，並且有一樣的總時間，為了做最後的平手判定，我們需要使用到最後 CORRECT 的 submission 時間。換句話說，如果 A 跟 B 都解了五題問

題，總時間都是 674，但是 A 的第五個過的 submission 比 B 還要早，那 A 的名次將會比 B 前面。

更準確的說，我們會透過比對兩份 submission 的「 submission id 」來當作上傳時間的比對，這是因為 id 越小的即代表越早上傳，而不可能有兩份 id 一模一樣的 submission 被記錄著。

5.5 罰時的例外

有兩種錯誤的 submission 不會造成罰時: COMPILER-ERROR 和 TOO-LATE。(請見可能的裁判結果)

5.6 計分板

在記分板上你會看到白色、綠色、深綠色、紅色或藍色五種格子。

- 「 白色 」表示你尚未上傳該題題目
- 「 綠色 」表示你解決了該題問題
- 「 深綠色 」表示你解決了該題問題，而且還是所有參賽選手中最早解出來的！
- 「 紅色 」表示你上傳了該題問題不過所有的 submission 都錯了
- 「 藍色 」表示你該題有個 submission 正在等待裁判結果

格子中第一個數字代表你傳了多少個 submission 到這題，如果是 CORRECT 的 submission (綠色的格子)，第二個數字代表你在比賽第幾分鐘通過了這題。

5.7 計分板凍結

在比賽的最後一個小時，所有的 submission 會在計分板上以藍色顯示而不會顯示結果，這些格子不會在比賽結束前變成綠色或是紅色。儘管你不會在最後一小時的計分板上看到自己隊伍的狀態，你還是可以回到 Home 來看到自己 submission 的裁判結果。

6 常見問題

6.1 General

6.1.1 我找到了個 bug？/ 我有個在文件中沒有答案的問題。

請聯絡我們，告訴我們這件事。

6.1.2 你們有存下我們的 submission 嗎？

有的，我們有存下你的 submission。偶爾有些題目會有問題，或是執行時間限制被修改了（這不會很常發生），我們會需要重新 judge 該題目的所有 submission。存下的 submission 也會用來偵測作弊。

6.2 Judging

6.2.1 Judge 是怎樣進行的呢？

請見程式評測過程。

6.2.2 我拿到了 TIMELIMIT，這表示我的輸出結果都是正確的嗎？

不是。我們會等到你的程式正常結束才檢查輸出的結果。

6.2.3 我拿到了 TIMELIMIT，你可以告訴我我的程式比時限慢了多少嗎？

對不起，沒辦法。如果你的程式跑超過時限，DOMjudge 會自動中止該程式，所以無法分辨你的程式是卡在無窮迴圈，或「只是慢了一些」。

6.2.4 如果有多組測試資料，執行時間上限的意思是？

如果有多組測試資料，執行時間上限表示「每組」測資都有這樣的時限。

6.2.5 我拿到 WRONG-ANSWER，這表示我的程式至少沒有因為你們的測試資料而 crash 嗎？

不，不代表喔。如果有多組測資 WRONG-ANSWER 有可能會在你程式跑過所有測資之前就偵測到。

6.3 C/C++

6.3.1 如果程式沒有 `return 0` 會怎樣?

如果程式回傳非零的值會拿到 RUN-ERROR，但現在的 C++ 或是 C99 的編譯器，只要

- 讓 main function 回傳值是 `int`，和
- 不要自行寫出回傳非零的值的程式

就不會有這樣的問題。

如果在 C 裡自行定義 main 的回傳型態，例如我們使用 `float` 或是 `void` 當作回傳的型態，會有未定義的行為，有可能會造成 RUN-ERROR。最後，絕對不會因為在 main 最後加上 `return 0` 而造成錯誤，所以如果順手的話就加上吧。

6.3.2 如果遇到使用 `cin` 和 `cout` 處理 I/O 速度太慢的話，該怎麼辦呢？

至少有三種以上的方法可以讓 `cin` 和 `cout` I/O 在 C++ 中的速度變快。

1. `cin` 和 `cout` 會要跟 C stdio 函式同步，我們可以透過使用 `ios::sync_with_stdio(false);` 來關掉這個同步。
2. `cin` 跟 `cout` 是綁定的，這表示從當你使用 `cin` 的同時，`cout` 會被 flush。當題目有大量輸入輸出交替的情況，這會讓速度變慢。為了要取消這個功能，我們可以使用 `cin.tie(NULL);` 指令，請注意 `sync_with_stdio()` 會 reset 這個設定，所以請確定這兩個指令的使用順序是對的。
3. 我們要知道 `cout << endl` 會 flush `cout`。但要注意的是這功能無法被取消，所以如果你要輸出大量的輸出，可能不要使用 `endl`，改用 `"\n"` 會好些。

7 聯絡我們

在比賽進行中，當對題目或是有其他問題的時候，請使用 Clarifications 功能聯繫裁判。如果是使用的電腦、列印、無法登入 DOMjudge 等等系統相關的問題，實體進行比賽時請洽周圍的工作人員，非實體或是比賽進行期間則可以使用此處連結內寫的聯絡方式聯絡我們。

8 外部連結

- NPSC 官方網站
- DOMjudge 官方網站
- DOMjudge 官方使用手冊（英文）
- C++ 文件

A 程式碼範例

本節作為範例來解決「北極熊大遷徙」這道題目。

「北極熊大遷徙」的題目簡述內容如下：輸入兩個整數 $a, b (1 \leq a, b < 2^{31})$，輸出 $a + b$ 的結果於一行。

以下是這題的一筆範例測試資料：

Input	Output
24 47	71

以下是一道用 C 撰寫的範例解答：

```c
#include <cstdio>

int main() {
    long long a, b;
    scanf("%lld %lld", &a, &b);
    printf("%lld\n", a + b);
}
```

以下是一道用 C++ 撰寫的範例解答：

```cpp
#include <iostream>

int main() {
    long long a, b;
    std::cin >> a >> b;
    std::cout << a + b << "\n";
}
```