# South African Programming Olympiad

Round 2 – CMS Evaluator Manual v2

21 April 2019

## Contents


**Getting Started** ..... 2

Registration ..... 2

Overview ..... 4

Communication ..... 4

Statement ..... 5

Submissions ..... 5

Documentation ..... 6

**Submitting solutions** ..... 6

C / C++ ..... 6

Java ..... 7

Delphi / Pascal ..... 7

Python 2 / Python 3 ..... 8

Scratch ..... 9

Template files ..... 10

**Evaluation** ..... 10

**Feedback** ..... 11

Live Contest ..... 11

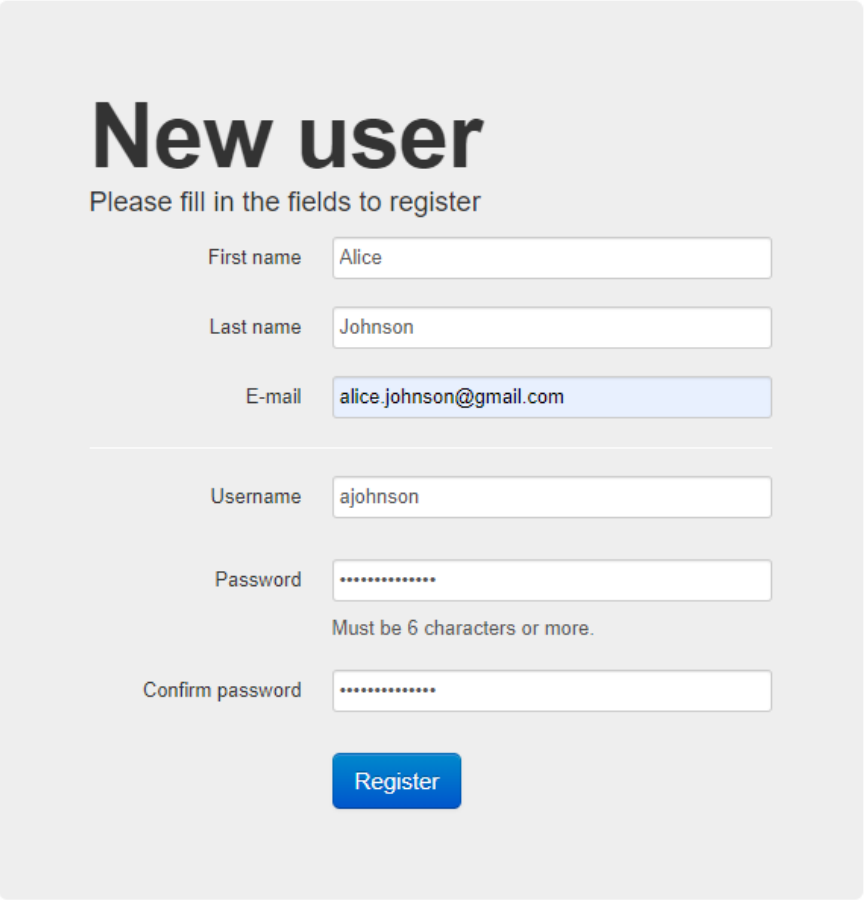
Post Contest ..... 11

## Getting Started

The evaluation server of the South African Programming Olympiad is hosted at https://saco-evaluator.tk/ . Visiting this page will provide numerous links to past problems, presentations, and other Olympiad resources which will help you prepare for the types of problems you will encounter in the Programming Olympiad.

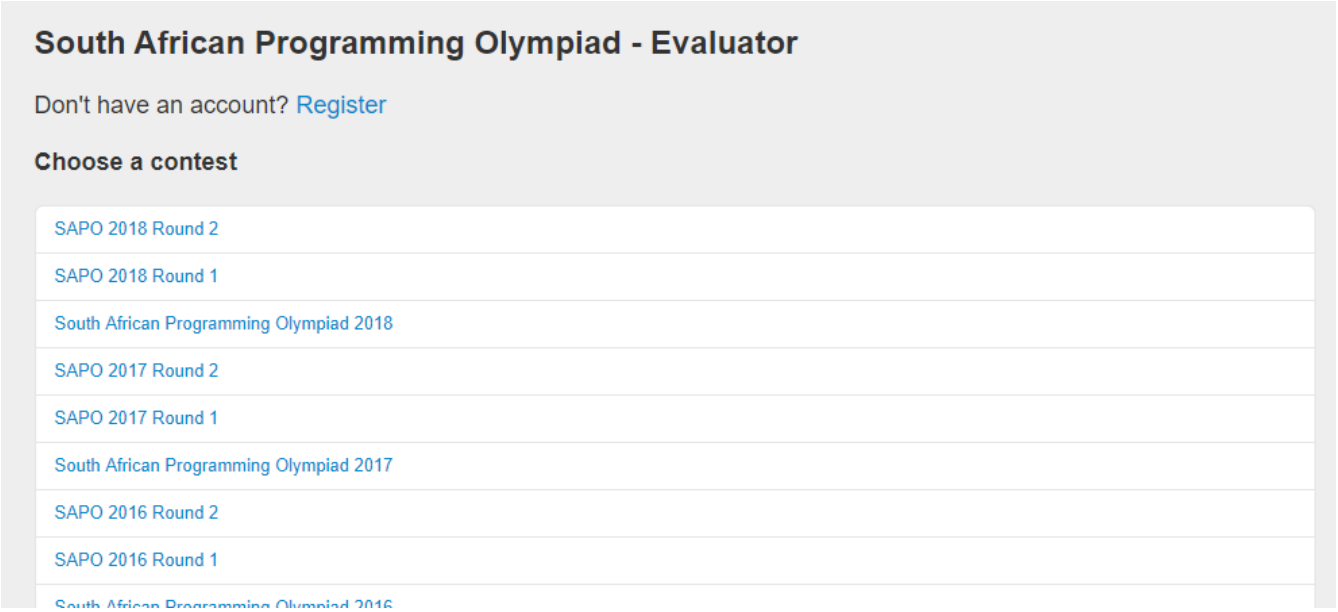
To get started with the online evaluator, go to https://saco-evaluator.tk/cms/ This link will be also be available on the homepage as a button with the text "**Enter Contest Server**".

Here, you will find a list of all past SAPO final round contests from 2002, as well as Round 1 and 2 contests from the past four years. Click on any contest, and then click **Register** to make a free account. Once you've filled in your details and clicked Register, a new account will be made with your provided username and password.

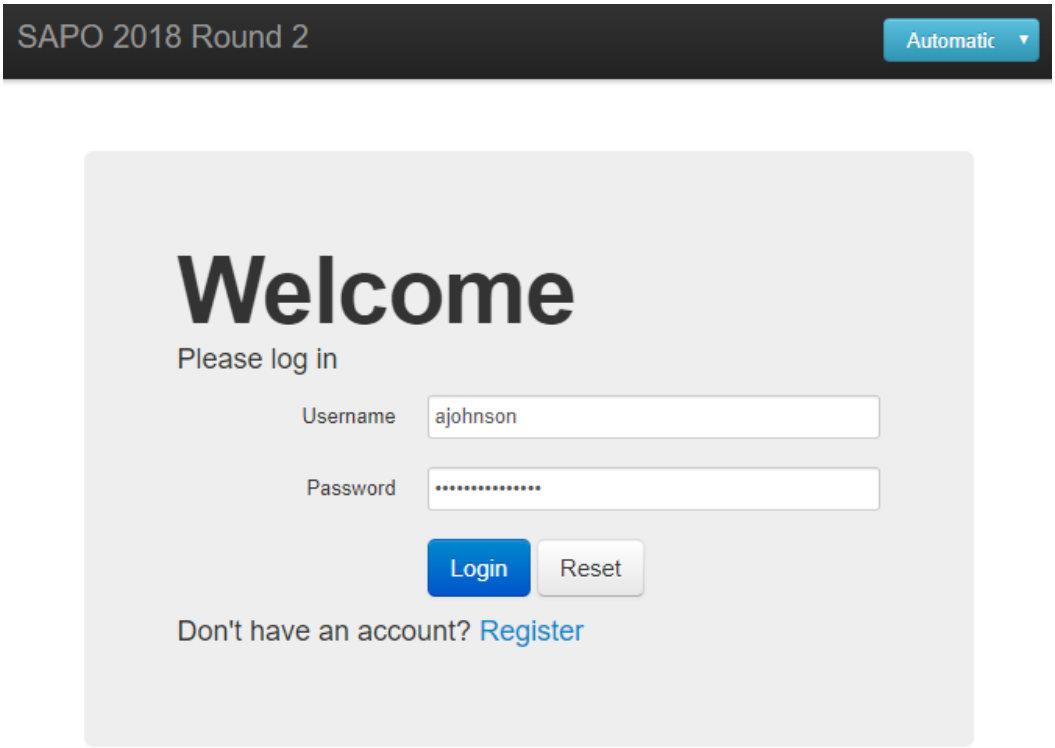



This account can be used to access any of the available contests shown on the contest page, with the exception of some internal training camp contests used for IOI selection.

From the contest list page at https://saco-evaluator.tk/cms/ , you can now select any of the past SAPO contests.

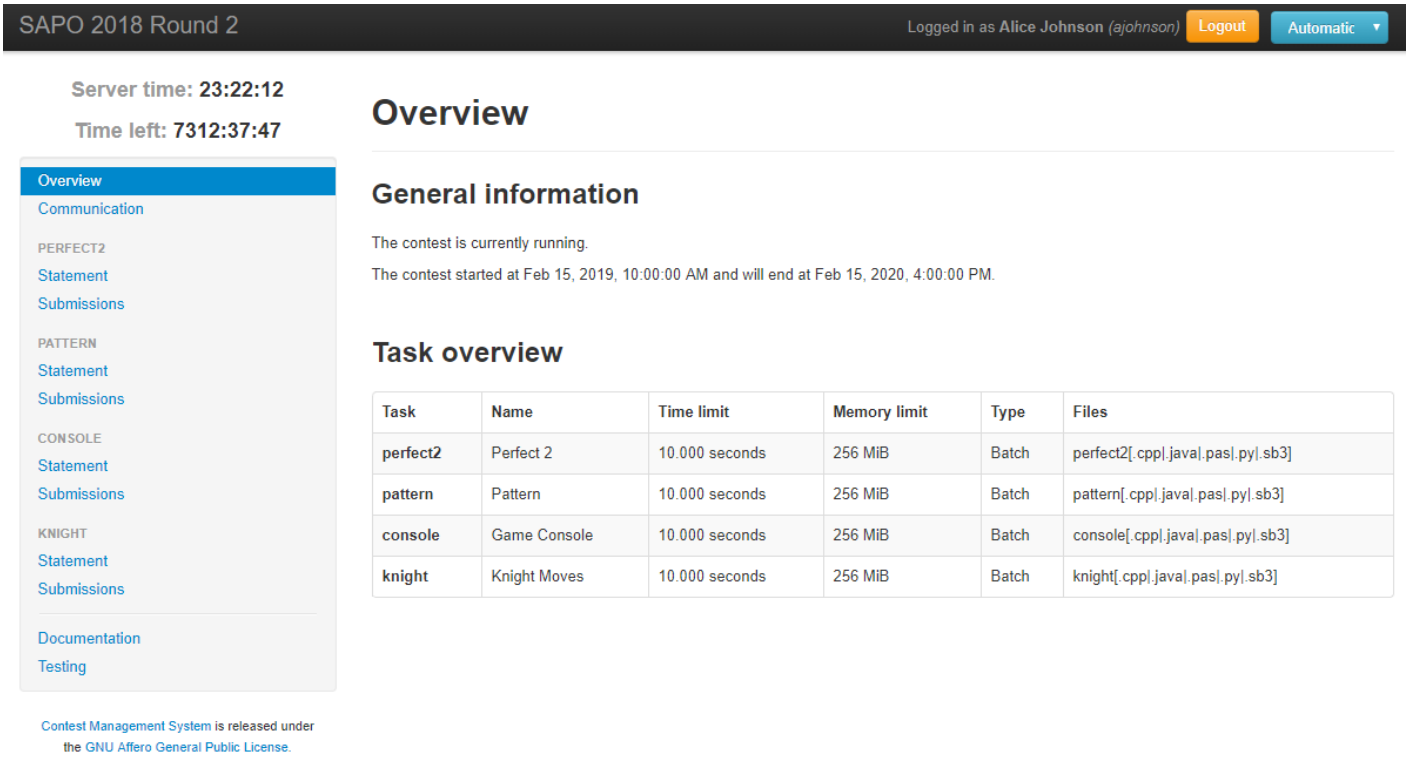



Click on any of the available contests. You will then see a log-in screen:




By default, the language that CMS uses for its user interface is English. This can be changed by clicking the blue button at the top-right and selecting your preferred language. At the moment, the only local languages supported are English and Afrikaans.

Fill in your username and password, and click **Login** to go the Overview page for the contest you selected:

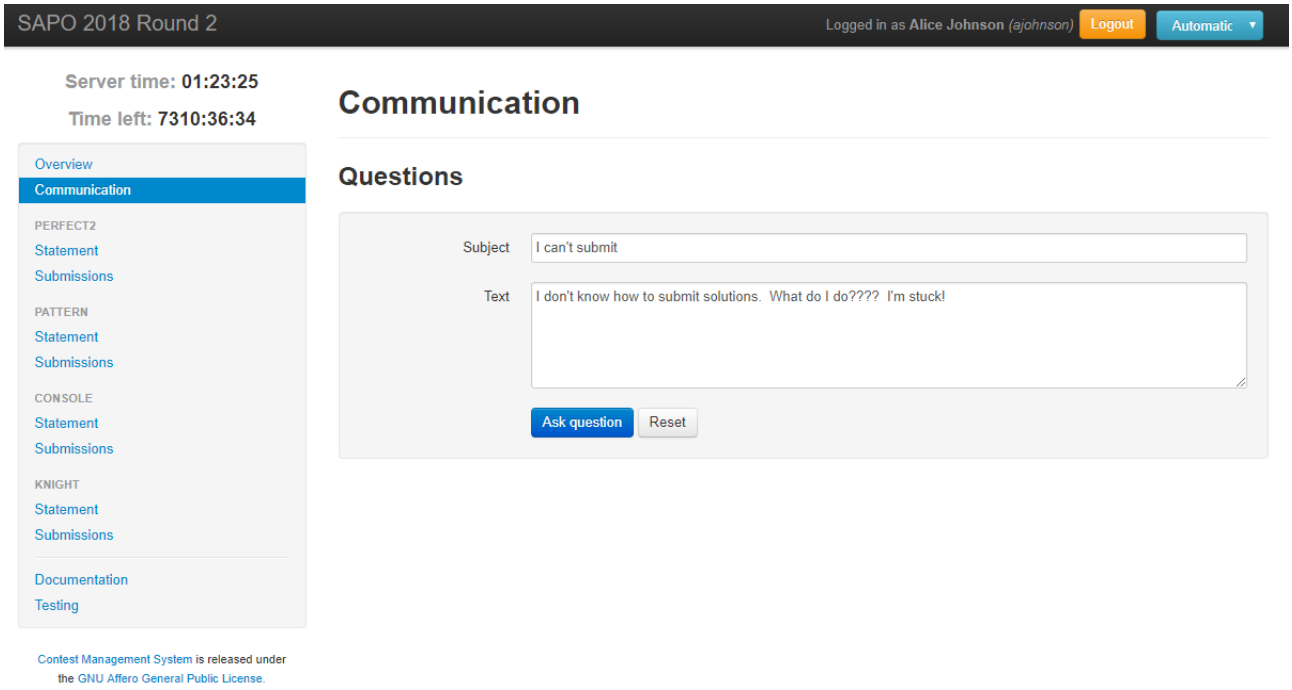



Here, you will find some general information about the contest you selected. Underneath **Overview**, the contest start and end time will be displayed. You can submit solutions to problems for the contest only after the start time and before the end time.

Beneath **Task overview**, will be a list of all problems for that contest. Here, you will find the problem name, time limit and memory limit. The time and memory limits give the largest amount of time and memory that your submissions may use up before the evaluator stops the execution of your code.

Several more options are given on the left-hand side of the screen.

## Communication

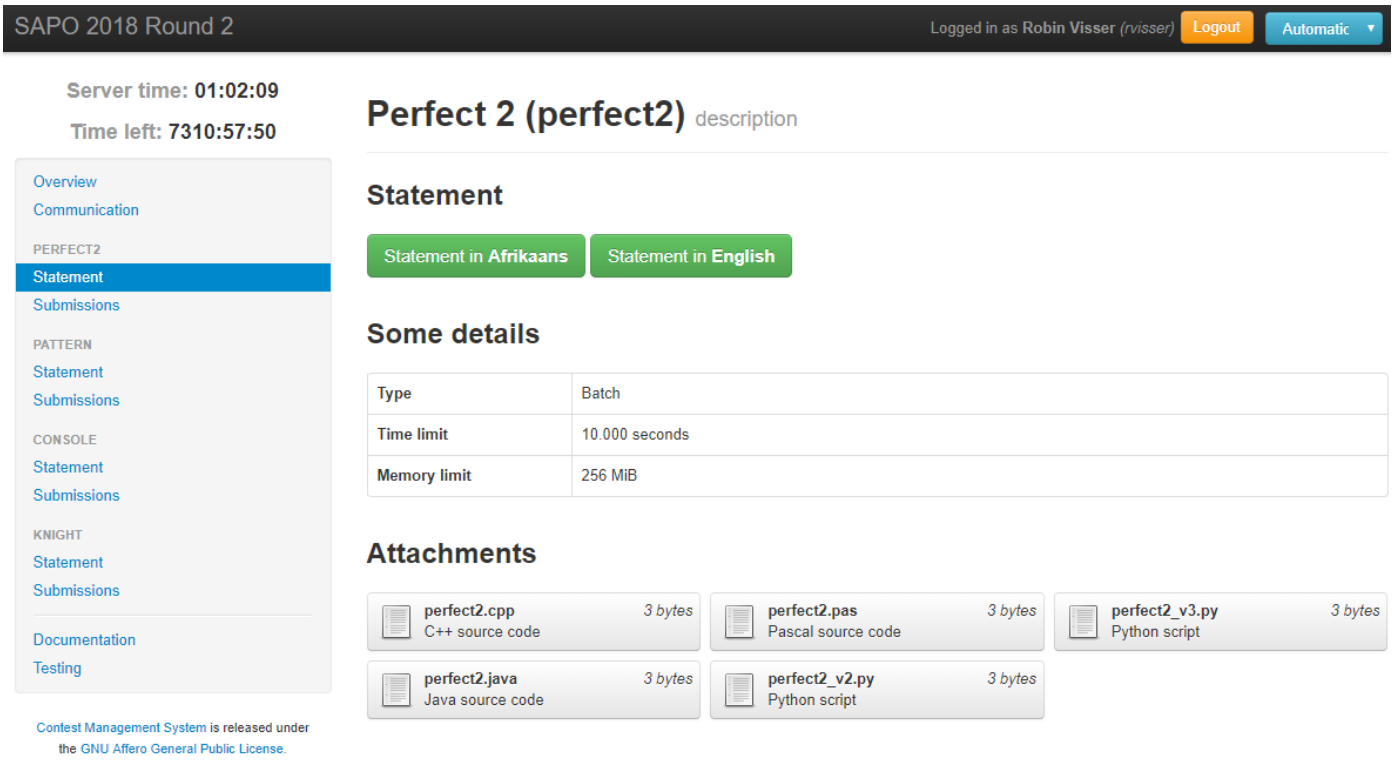


If you wish to ask any question regarding any of the contest problems, or are having trouble using the online evaluator, you may submit a question to the judges using the form given. During live contests, a response will be given as soon as possible, otherwise, we will aim to respond within 24 hours.

For each problem of the contest, there is a **Statement** and **Submissions** page.

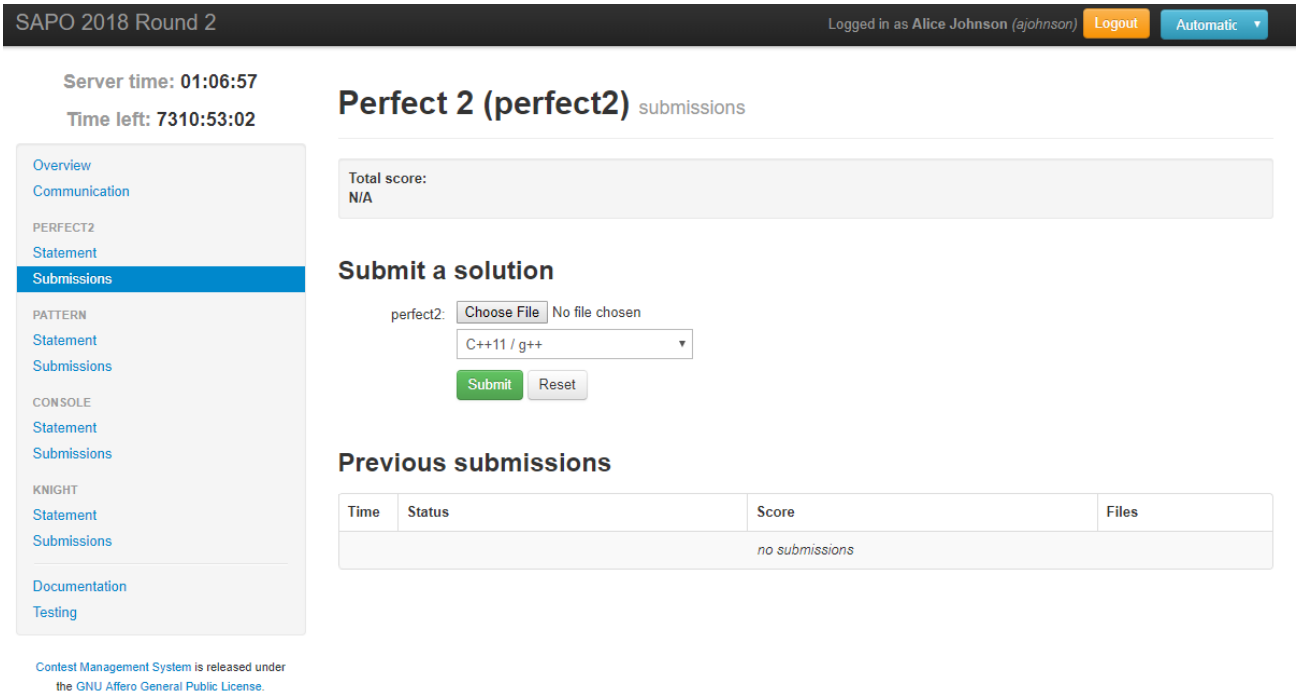
## Statement

Here, you can download the task statement of the problem in any of the available languages. The time and memory limits will also be given. Any attachments related to the problem can also be found here. This includes template files you can use to assist with your submissions.




## Submissions

This is where you submit your submissions to the contest problems. Simply click “Choose File” and select the source code file from your computer you wish to submit. The evaluator should automatically detect what language is used based on the file extension, otherwise select the language from the drop-down menu. Finally, click Submit to upload your submission to the evaluator.




Details of all your prior submissions can be found under "Previous submissions".

## Documentation

Here you'll find documentation for the languages supported during the contest. You may access any of these doc links during live contests, but not any other online resources. Beneath the documentation links are given detailed explanations for some of the messages you might see when submitting code.

## Submitting solutions

When coding up a solution to one of the contest problems, all input and output must be done using the standard input and output streams (**stdin** and **stdout**). This simply means that input and output is done the same way as using a console or terminal.

We shall illustrate this by using a simple example where the input asks for two integers $A$ and $B$, and the code must output the sum $A + B$. Examples of how to do this in each of the supported languages are given below:

### C / C++

The easiest way to do input/output in C++ is to use `cin` and `cout`.

```
#include <iostream>
using namespace std;

int main() {
    int a, b;
    cin >> a >> b;
    cout << a + b << endl;
}
```

Make sure to put `using namespace std;` before your main method, otherwise you'd have to always write `std::cin` and `std::cout` instead of just `cin` and `cout`.

Alternatively, if you're using only C, or just want to stick with using C syntax in C++, then you can use the `printf` and `scanf` functions.

```
#include <stdio.h>

int main() {
   int a, b;
   scanf("%d%d", &a, &b);
   printf("%d\n", a + b);
   return 0;
}
```

## Java

For Java, you'd want to use either `Scanner` or `BufferedReader`

```
import java.util.*;


public class sum {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int a, b;
        a = in.nextInt();
        b = in.nextInt();
        System.out.println(a+b);
    }
}
```

Using a BufferedReader is often much faster with taking in input than Scanner, however for Round 2, using either one would be fast enough.

## Delphi / Pascal

If you're using Delphi, you will need to use the Console mode and not use any graphical interfaces. To take in input and output, simply use the `readln` and `writeln` functions, as shown below:

```
var
   a, b: integer;

begin
   readln(a, b);
   writeln(a + b);
end.
```

Note that you may define any other functions or variables as you like. You can, however, always code everything within a single main program block using `begin` and `end.`. You cannot though use anything that requires a graphical interface.

You are allowed to use object-orientation. If you use packages such as `CustApp`, then you may need to submit your code in Delphi mode. This can be achieved by just putting `{$MODE Delphi}` near the top of your code.

The syntax between Delphi and Pascal is very similar, however there are some key differences. For a full list, see: http://wiki.freepascal.org/Code_Conversion_Guide#Syntax_differences

For further details on how to start a console application using Delphi software, consult the attached document "Creating A Console Application".

## Python 2 / Python 3

We support both Python 2 and Python 3 on the evaluator. You must however ensure that the correct version is selected when submitting a problem, since the evaluator will not automatically detect which version must be used.

For input/output, use the `input()` and `print()` functions. Note that `input()` takes input one full line at a time, so if several integers are given on the same line, one must separate out each integer individually after taking in input. The easiest way to achieve this is to use the `.split()` method. This will convert the line of input into a list of strings. To cast as an integer, simple use the `int()` function.

Note that whilst the two versions of Python are similar, there are some key differences. Most notably, Python 2 uses `raw_input()` to take in input as a string, whereas Python 3 uses `input()`. With Python 3, output must be done using `print()` (**with** brackets) whereas Python 2 does not require them. For a more comprehensive list, see: https://www.w3resource.com/python/python-2-vs-3.php

Examples of summing two numbers are given in three different ways for each version:

For Python 2:

```
inp = raw_input().split()
a = int(inp[0])
b = int(inp[1])
print a+b
```

OR

```
a, b = [int(i) for i in raw_input().split()]
print a+b
```

OR

```
a, b = map(int, raw_input().split())
print a+b
```

For Python 3:

```
inp = input().split()
a = int(inp[0])
b = int(inp[1])
print(a+b)
```

OR

```
a, b = [int(i) for i in input().split()]
print(a+b)
```

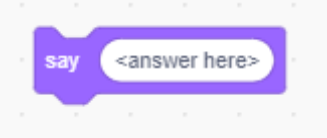
OR

```
a, b = map(int, input().split())
print(a+b)
```

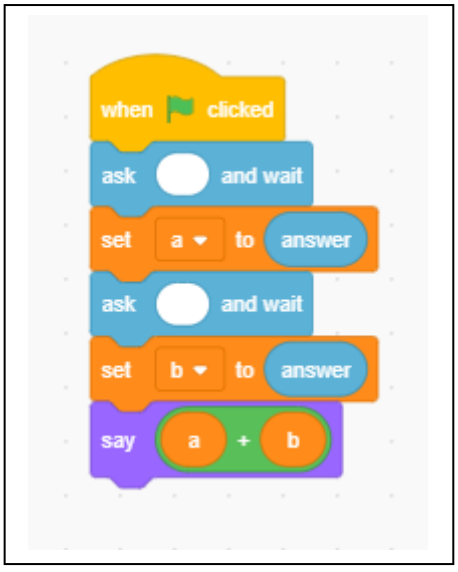
## Scratch

All versions of Scratch, including v1.4, v2.0 and v3.0 are supported on CMS. We don't guarantee however that all problems, especially the more difficult ones, have 100% solutions in Scratch. To take in input, simply use the `ask` block, **without** any prompts:


Output can be done in the usual way, using a `say` block:




An example of how one would add two given integers in Scratch is given below:

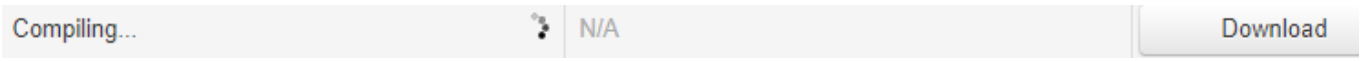


### Template files

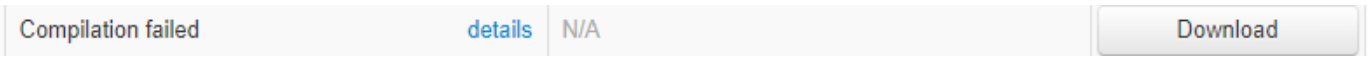
During the contest, we will provide template files for each program in each language which you can use to assist with input/output. There is no obligation to use them; you are welcome to submit entirely your own code. However, if you are having trouble taking in input and output correctly, then consider downloading the relevant template file (found under "**Attachments**" on the "**Statement**" page), and then simply add your own code in the main method.

## Evaluation

Once you've submitted code to a particular problem, the evaluator will automatically start compiling your code. The following will be displayed:

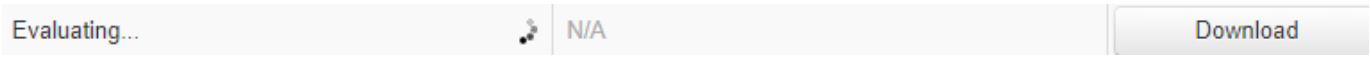



If compilation was not successful, you will receive a **Compilation failed** message, as shown below:




Further details on why the evaluator could not compile your code can be found by clicking on "details". Here, you will find more informative compilation logs which will help you debug your code.

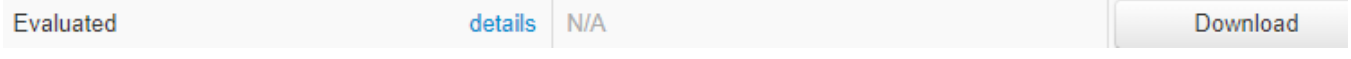
Otherwise, if your code compiles successfully, it will then start evaluating your code on the test data.




Depending on how busy the evaluation servers are, evaluation could take as quick as a few seconds to several minutes. Once evaluation is done, it will say "Evaluated", and a score will be allocated to your submission:


By default, the total score for the problem will simply be taken as the highest result you obtained across all your submissions. The evaluator will automatically keep the total score updated at the top of the submission page for that problem.

Note that during live contests, your score will not be shown, and instead will simply display N/A.



## Feedback

Once a submission has been evaluated, you will be able to see the outcome of your submission on some of the test cases. For each submission, click on "details" to see how it performed.

For each test case, there are several possible outcomes that may occur.

### Live Contest

During a live contest, you will only receive detailed feedback on the example test cases given in the task statement. No feedback will be provided on any hidden test cases during the contest.

For example, you might receive the following feedback:

**Submission details**

Subtask 1 **(0 / 0)**

| # | Outcome | Details | Execution time | Memory used |
|---|---|---|---|---|
| 1 | Correct | Correct | 0.028 sec | 4.05 MiB |
| 2 | Correct | Correct | 0.028 sec | 4.11 MiB |
| 3 | Not correct | Incorrect. 1st lines differ - expected: '2', found: '1' | 0.028 sec | 4.05 MiB |

Subtask 2 **(N/A)**

| # | Outcome | Details | Execution time | Memory used |
|---|---|---|---|---|
| 1 | N/A | | | |

This means that there were 3 example test cases given in the task statement. The submission produced the correct output for the first two test cases, however, it did not produce the correct output for the third test case. Specifically, the official solution expected your code to produce an answer of 2, whereas the output of your code was 1. Note that no information is given for any further test cases. You must therefore ensure your code produces the correct output for any valid input and not just the given example test cases.

### Post Contest

After the contest, full feedback will be enabled. You will therefore be able to see your exact score for each problem

**Submission details**

Subtask 1 **(0 / 0)**

| # | Outcome | Details | Execution time | Memory used |
|---|---|---|---|---|
| 1 | Correct | Correct | 0.028 sec | 4.03 MiB |
| 2 | Correct | Correct | 0.027 sec | 4.11 MiB |
| 3 | Not correct | Incorrect. 1st lines differ - expected: '2', found: '1' | 0.028 sec | 4.05 MiB |

Subtask 2 **(0 / 100)**

| # | Outcome | Details | Execution time | Memory used |
|---|---|---|---|---|
| 1 | Correct | Correct | 0.027 sec | 3.99 MiB |
| 2 | Not correct | Incorrect. 1st lines differ - expected: '5', found: '1' | 0.028 sec | 4.07 MiB |
| 3 | Not correct | Incorrect. 1st lines differ - expected: '14', found: '1' | 0.027 sec | 4.05 MiB |
| 4 | Not correct | Incorrect. 1st lines differ - expected: '59', found: '1' | 0.027 sec | 4.01 MiB |

Here, you will be able to see the full detailed feedback for each test case. In the above example, the code did not seem to do very well. Indeed, all the code did was output 1 for every test case and therefore got most of the answers wrong.

There are several other possible responses that can be obtained. *Execution timed out* simply means that your code did not terminate within the given time limit. You might also see *Execution killed* or *Execution failed because the return code was nonzero.* This usually means your submission crashed in some way. This could result from several possibilities; perhaps you tried to divide by 0, or accessed an array out of bounds, or used a string as if it were an integer etc.

In general, you are responsible for debugging your own code and ensuring that it solves the problem generally for all possible inputs.

---

If you have any further questions regarding usage of the evaluator, feel free to send us a message on the evaluator itself, or to contact us at saco.evaluator@gmail.com The evaluation server is always continuously undergoing development and we would certainly appreciate any feedback and suggestions on improvements that can be made.