

Part I: About the contest

Rules and Regulations

1. This is a timed contest.
2. Mobile phones will not be allowed in the contest. The contestant will submit them and will be able to retrieve them at the end of the given round/competition.
3. Use of the internet is strictly prohibited.
4. For any doubt of dishonesty, the contest organizers reserve the right to cancel the contest for the given contestant team.

Contest Organization Method:

1. The contest will be taken on the PC2 (Programming Contest Control) system. No internet will be provided except for the pre-configured connection of the app with the servers.
2. A contestant team needs to pass 90% of the test cases to proceed with the validator.
3. If two teams have the same score, the one with earlier submission(s) will be considered in the lead.
4. VS Code will be provided as a code editor.
5. At the limit, the clock will stop, and server will be closed for further evaluations.
6. The timeout provided for each problem will be a maximum of 10 seconds. This varies from problem to problem.
7. You must submit the problem without using any third-party libraries (*NumPy etc. in python*)
8. In the entire time, you have a liberty of submitting whatever problem you want to at whatever time. The entire time is at your disposal.
9. You will be allowed submissions in following languages:
 1. Python
 2. GNU C++/ C
 3. Microsoft C#
 4. Java
10. For your help, a sample problem is being shared, and how to operate the PC2 system regarding that.

Sample Problem: *List within Lists*

“For a given list of integers nested till n th number, implement a function that returns the sum of all these lists as if they were represented by a flattened one-dimensional list. You cannot use the flat- function.”

Input format:

A single line input is given, which contains a list (*entries delimited by spaces*) as a string. The brackets of the string will be in standard C/Python format. The program will read the string by standard input, parse it into the n -dimensional list, and then perform the required computation. A sample of input format is given:

```
[1 2 [3 4 [5 6] 7] 8 [9]]
```

This input, when performed the required computation, will produce the output of `45`

Constraints:

```
1 <= n < 6
```

Part II: How to Use PC2 to submit the solution:

This part of the documentation covers how to use the PC2 software to submit the problem. This part has two modules: 1. How to use PC2 Client, and 2. How to use PC2 Web App.

Using the PC2 client

1. At the start of your contest, you will be provided with login credentials to log into the client.
2. After logging in, you should see something of the sort:

PC^2 TEAM 1 (Site 1) [RUNNING] 9.8.0-6456

2:11 Exit

Submit Run View Runs Request Clarification View Clarifications Options About

Problem
Select Problem ▼

Language
Select Language ▼

Main File Select

Additional Files

Add Remove

Test Submit

3. For submitting the problem code, you just implemented, head over to the first dropdown and select the problem name from the available problems.

PC^2 TEAM 1 (Site 1) [STOPPED] 9.8.0-6456

2:10 Exit

Submit Run View Runs Request Clarification View Clarifications Options About

Problem
Select Problem ▼

Select Problem
Icecreams

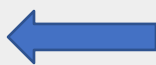
Select Language ▼

Main File Select

Additional Files

Add Remove

Test Submit



4. After it is selected, select the language you coded that problem in.

PC^2 TEAM 1 (Site 1) [STOPPED] 9.8.0-6456

2:09

Exit

Submit Run View Runs Request Clarification View Clarifications Options About

Problem
Select Problem

Language
Select Language

Python
Java
GNU C++
C#

Select

Additional Files

Add Remove

Test Submit

5. Now, select the problem file code you coded on the pc. For this, head on the third 'Main' and the 'select' button. This should open the directory navigator.

PC^2 TEAM 1 (Site 1) [STOPPED] 9.8.0-6456

2:08

Exit

Submit Run View Runs Request Clarification View Clarifications Options About

Problem
Select Problem

Language
Select Language

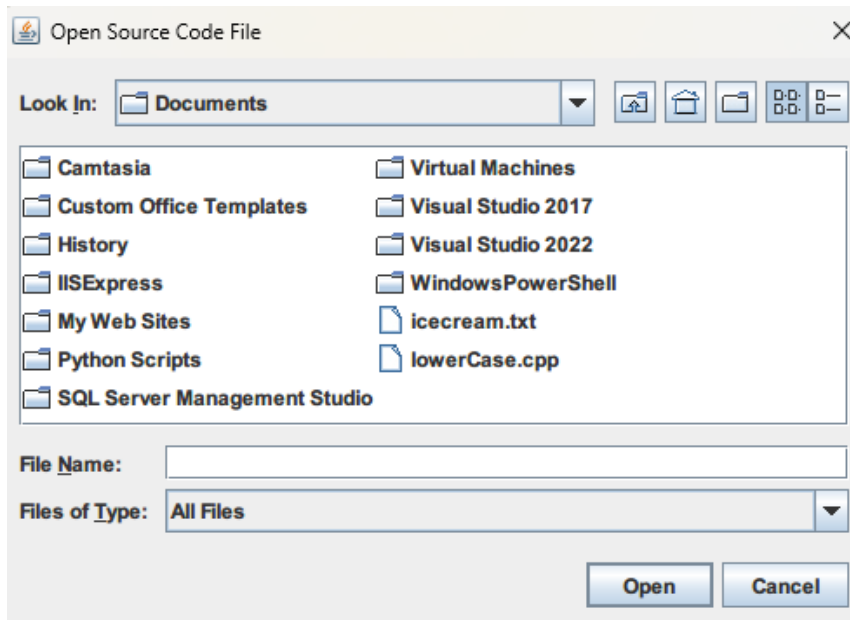
Main File

Select

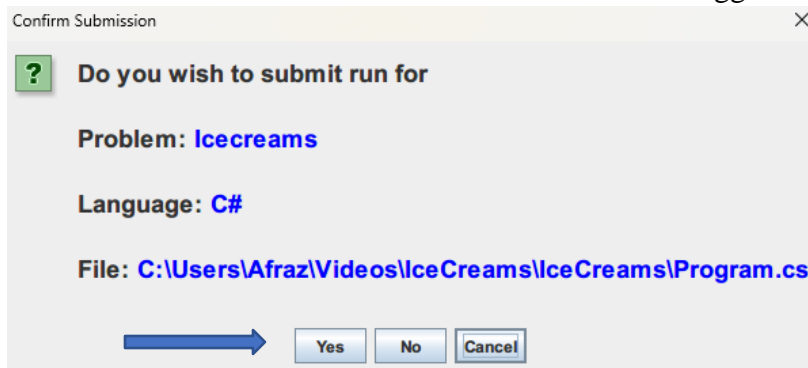
Additional Files

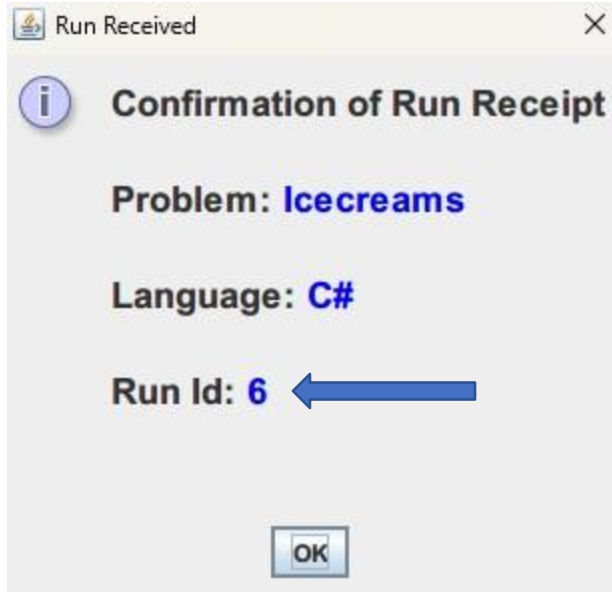
Add Remove

Test Submit



6. After the file is selected, add additional files as per necessary. Most of the time, however they aren't needed. More information about the problem will be given in the problem manual regarding this.
7. If available for the given problem, you may run your code to see what output it is generating. However, the problem itself will be graded on hidden test cases.
8. After submitting a particular problem, a run ID will be assigned to you. Let the contest administrator know about this run id so that it can be toggled a response for evaluation.





Let the contest administrator know about this run ID. Promptly, a response will arrive letting you know the validator.

Using the PC2 Web App

Using the PC2 Web App is something more easily understandable, and in fact, the method of choice of contest running will be this. This part covers how to use that.

1. Log onto the screen already loaded with your credentials.
2. You should see the screen in front of you after logging into the screen (called the **Runs** screen). You can see the Runs Screen in *Figure 1* on the next page.
3. After logging into Runs, you will want to submit the problem you just coded a solution to. For this, click the “Submit Problem” Button on the top left of the Runs Screen. A prompt of the sort in *Figure 2* should appear in front of you. Fill in the details (select problem, language, and files) and submit the problem. As in previous example, you will receive a run ID. This run will be logged in the dashboard on the Runs Screen in front of you. A sample of the runs screen after a submission is shown in *Figure 3*.
4. Depending on contest configuration, you will receive a pop up describing the status of your run. If you don't, don't fret. In the Run Logs as shown in *Figure 3*, the status of your run will be updated to whatever the result was (If your code passed then you will see yes, else no).

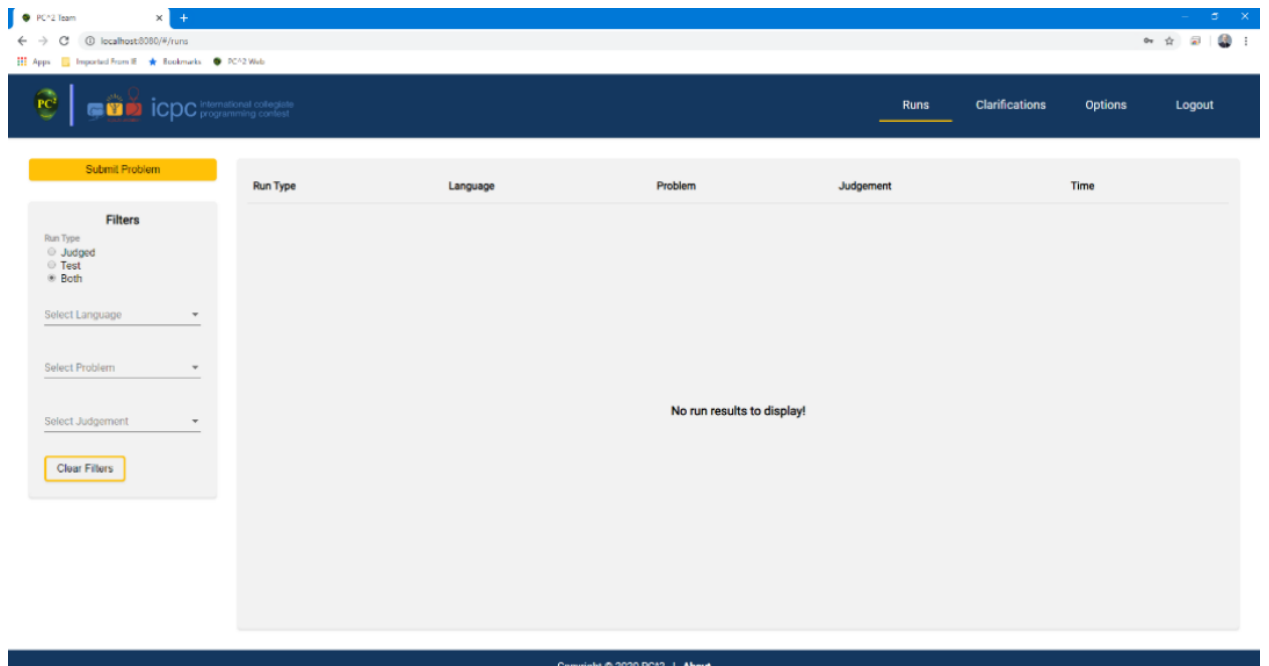


Figure 1. Runs Screen

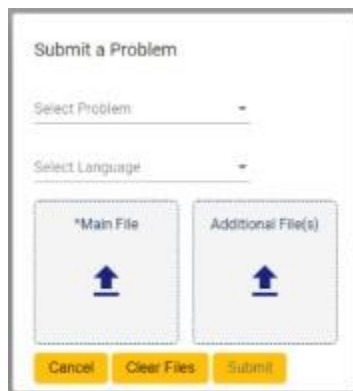


Figure 2: Submit Problem prompt.

The screenshot shows the ICPC Run Logs interface. On the left is a sidebar with filters. The main area displays a table of submission results.

Run Type	Language	Problem	Judgement	Time
Judged	Python	Evergreen	Yes (Preliminary)	74
Judged	C++	Dogweed	Yes	59
Judged	Java	Alphabets	Yes	56
Judged	Java	Dogweed	Wrong Answer	57
Judged	Python	Carhopping	Yes	55
Judged	Java	Alphabets	No - Run-Time Error	53
Judged	Java	Bowling	Yes	51
Judged	C++	Alphabets	No - Compilation Error	48

Figure 3: Run Logs after submission.

Part III: About Submissions

1. Against one problem, multiple submission are allowed. However, the earliest submitted, most successful one of them will be considered.
2. For a problem, if a solution is submitted, it will remain in the problems prompt. This is for the above-mentioned point.
3. Results will be made known to the contestants for proceedings in the next round(s).

Appendix: Right and Wrong ‘Problems’

This appendix deals with the submission of right and wrong problems. For reference, we will take the sample problem mentioned in Part 1 of this manual: List within Lists. The solution to that problem will only show you the idea behind ‘*what constitutes a wrong submission*’.

Let us consider the criteria for judgement, most chiefly is the ‘**Output Validator**’. This is an extremely important point: Your submission is literally compared word-by-word, case-by-case, space-by-space with the test case output. If even a slight mismatch occurs, the test case will be considered wrong in its entirety. This mismatch also extends to outputting on the console.

What is meant in the above paragraph, is the fact that if something is asked for output in the console validator, only that thing is required. No extra message, no nothing.


Let us see a practical example. We will show the code in Python, you must get the gist behind it.

Right submission:

```
def ListSolver(nested_list):
    total_sum = 0
    for item in nested_list:
        if isinstance(item, int):
            total_sum += item
        elif isinstance(item, list):
            total_sum += ListSolver(item)
    return total_sum

input_str = input()
# Remove all whitespace characters from the input string
input_str = input_str.replace(" ", "").replace("\n", "")
# Convert the input string to a nested list of integers
nested_list = []
stack = [nested_list]
for char in input_str:
    if char == "[":
        new_list = []
        stack[-1].append(new_list)
        stack.append(new_list)
    elif char == "]":
        stack.pop()
    else:
        stack[-1].append(int(char))

# Call the ListSolver function with the nested list and print the result
result = ListSolver(nested_list)
print(result)
```



Notice the last line of this code. There is only one print statement in this entire code, and even that is printing a number only that is returned. So, this code is considered fundamentally correct, for it will output only '45' and nothing else to the test case for the sample problem.


Wrong submissions:

Now let us consider the very same code, but with a small modification:

```
def ListSolver(nested_list):
    total_sum = 0
    for item in nested_list:
        if isinstance(item, int):
            total_sum += item
        elif isinstance(item, list):
            total_sum += ListSolver(item)
    return total_sum

input_str = input()
# Remove all whitespace characters from the input string
input_str = input_str.replace(" ", "").replace("\n", "")
# Convert the input string to a nested list of integers
nested_list = []
stack = [nested_list]
for char in input_str:
    if char == "[":
        new_list = []
        stack[-1].append(new_list)
        stack.append(new_list)
    elif char == "]":
        stack.pop()
    else:
        stack[-1].append(int(char))

# Call the ListSolver function with the nested list and print the result
result = ListSolver(nested_list)
print('Result is ' + str(result))
```



In the last line, notice the subtle change. Because, if we look closely, the output now obtained on the console will be 'Result is 45' for the test case of the problem instead of plain '45'. It might seem OK to the human eye but for PC2 system, the concept is vastly different.

The same thought extends to outputting 'Enter input:', 'The Result is', *etc. etc.* on the console for taking input.