

AIPO Tips and Tricks

Andrew Nash

Feb 2023

Introduction

The first problem in the 2024 preliminary round is a very simple summation problem. The intention of this is to make sure that you understand the format that solutions are expected to be in. If you are new to competitive programming, and are struggling with this aspect, this document is a guide to help explain some common pitfalls specific to how our server accepts code submissions.

Simple Sample Problem

You are given three lines of input:

Line 1 Two space separated integers, a and b

Line 2 A single integer n

Line 3 n space separated integers

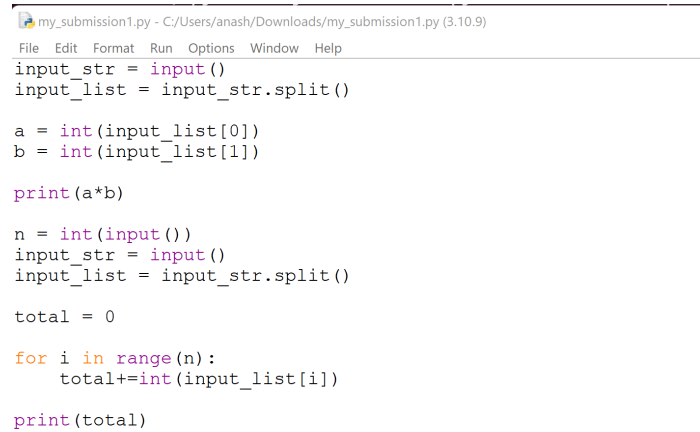
Your task is to output, on two lines, the product $a * b$, and the sum of the line of n integers, i.e. $\sum_{i=1}^n L_i$

Solving the problem

A critical aspect of solving any problem is to perform inputs, and present outputs in the correct manner.

When grading submissions, the server will **evaluate** your submission against **test cases**. For each test case, the sample input is passed into your program via the console. Your program's output is captured, and compared to a the test case solution. Any difference in output results in that test case being a **fail**, otherwise it is considered a **pass**.

Lets assume that you have a program, which is a solution to the above problem. Don't worry if you don't understand fully everything that this code is doing, this will be discussed later.

A screenshot of a Python script in a text editor. The title bar shows 'my_submission1.py - C:/Users/anash/Downloads/my_submission1.py (3.10.9)'. The menu bar includes 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code is as follows:

```
input_str = input()
input_list = input_str.split()

a = int(input_list[0])
b = int(input_list[1])

print(a*b)

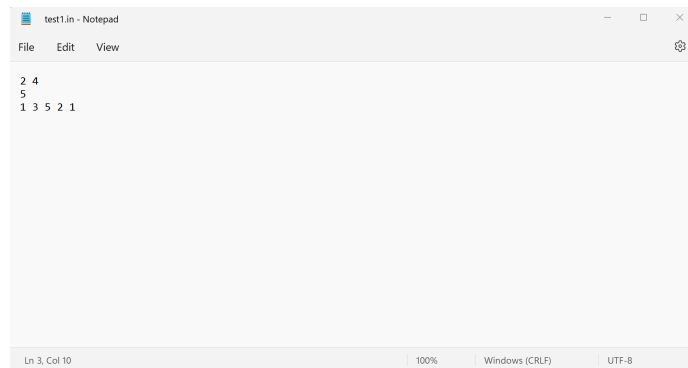
n = int(input())
input_str = input()
input_list = input_str.split()

total = 0
for i in range(n):
    total+=int(input_list[i])

print(total)
```

Figure 1: A correct solution

Lets consider the example of one test case. In reality, there will be many test cases for each problem.

A screenshot of a Notepad window titled 'test1.in - Notepad'. The menu bar shows 'File', 'Edit', and 'View'. The text content is:

```
2 4
5
1 3 5 2 1
```

The status bar at the bottom indicates 'Ln 3, Col 10', '100%', 'Windows (CRLF)', and 'UTF-8'.

Figure 2: A test case input

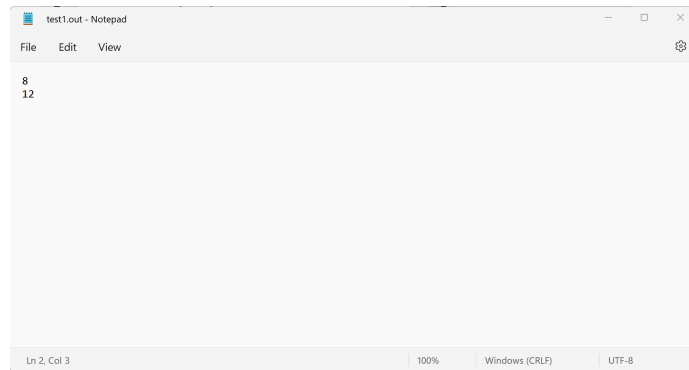


Figure 3: A test case expected output

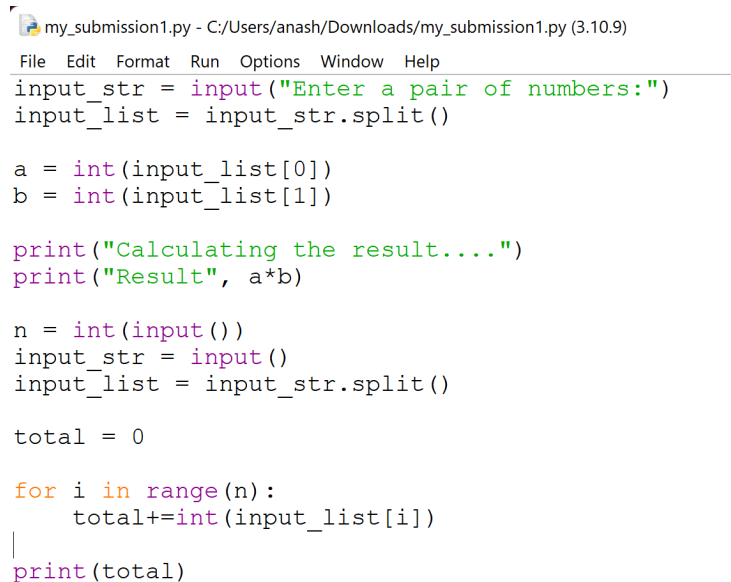
The server will, under the hood, do something similar to the following:

```
anash@LAPTOP-KJGTFHHJ:/mnt/c/users/anash/Downloads$ cat test1.in | python3 my_submission1.py
8
12
anash@LAPTOP-KJGTFHHJ:/mnt/c/users/anash/Downloads$ cat test1.in | python3 my_submission1.py > submission_output.out
anash@LAPTOP-KJGTFHHJ:/mnt/c/users/anash/Downloads$ diff test1.out submission_output.out
anash@LAPTOP-KJGTFHHJ:/mnt/c/users/anash/Downloads$
```

Figure 4: How the server will grade a submission on a test case

Here, we see that the *diff* command doesn't return anything - so the server considers this test case a pass, and will award points for this test case. If there are ten test cases for this problem, passing this case would be worth 10 points.

If however, you submit this code, which does the same thing as the above code, but contains extra `input()` and `print()` statements, the test case will fail



```

my_submission1.py - C:/Users/anash/Downloads/my_submission1.py (3.10.9)
File Edit Format Run Options Window Help
input_str = input("Enter a pair of numbers:")
input_list = input_str.split()

a = int(input_list[0])
b = int(input_list[1])

print("Calculating the result...")
print("Result", a*b)

n = int(input())
input_str = input()
input_list = input_str.split()

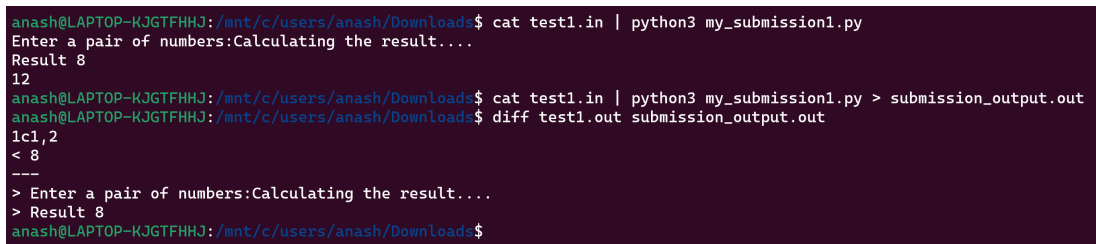
total = 0

for i in range(n):
    total+=int(input_list[i])
|
print(total)

```

Figure 5: An incorrect submission

Look at what happens when the server tries to grade this problem:



```

anash@LAPTOP-KJGTFHHJ:/mnt/c/users/anash/Downloads$ cat test1.in | python3 my_submission1.py
Enter a pair of numbers:Calculating the result...
Result 8
12
anash@LAPTOP-KJGTFHHJ:/mnt/c/users/anash/Downloads$ cat test1.in | python3 my_submission1.py > submission_output.out
anash@LAPTOP-KJGTFHHJ:/mnt/c/users/anash/Downloads$ diff test1.out submission_output.out
1c1,2
< 8
---
> Enter a pair of numbers:Calculating the result...
> Result 8
anash@LAPTOP-KJGTFHHJ:/mnt/c/users/anash/Downloads$

```

Figure 6: The arguments to input(), and extra print statements cause the diff command to find differences in the files that cause a fail

The lesson here is that you should **NEVER** include any unnecessary print statements, prompts in input() commands. Further, the layout of your inputs and outputs are important. If the problem specifies that an input is one or multiple lines, your code must account for this precisely.

Submitting the problem to the server

First, log in to the server, as per the instructions in the registration form. Navigate to the particular problem you want to make a submission for.

TestContest

Logged in as Test User (Dummy) Logout Automatic

Server time:
21:29:00

Time left:
60290:30:59

Overview

Communication

SIMPLESUMS

Statement

Submissions

Documentation

Testing

Contest Management System

is released under the GNU

Affero General Public

License.

SimpleSums (SimpleSums) submissions

Score:
0 / 100

Submit a solution

SimpleSums: No file chosen

C++11 / g++

Previous submissions

Time	Status	Score	Files
no submissions			

Figure 7: The submissions page

From here you can upload your submission, select the language used, and submit it for grading.

TestContest
Logged in as Test User (Dummy)
Logout
Automatic

Server time:
21:29:33
Time left:
60290:30:26

Overview
Communication
SIMPLESUMS
Statement
Submissions
Documentation
Testing

Contest Management System
is released under the GNU
Affero General Public
License.

SimpleSums (SimpleSums) submissions

Score:
0 / 100

Submit a solution

SimpleSums:
 my_submission1.py
Python 3.9.12 / PyPy 7.3.9

Previous submissions

Time	Status	Score	Files
no submissions			

Figure 8: Upload a solution, making sure the language dropdown matches the language used to code the solution (Python or C++)

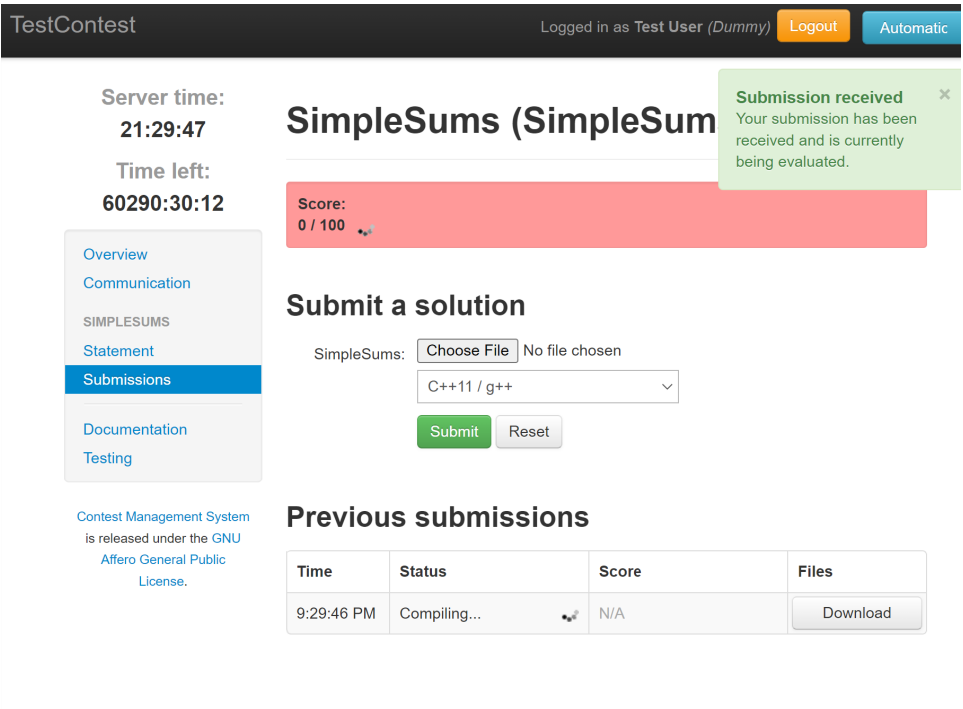


Figure 9: The server will take a little while to return a grade, please be patient when waiting for results.

When the server is finished grading your submission, you will be able to review your results on each testcase. In this case, there is only one testcase. If you upload the second, incorrect, solution with extra print statements above, for example, you will get an outcome of *WRONG ANSWER*. This verdict means that your solution ran, and printed an answer, but it did not match the correct output for the testcase. This may be caused by incorrect print statements, or potentially a logical issue in your code, where it is not *correct* - and is using an incorrect algorithm to solve the task.

Server time: 21:30:06

Time left: 60290:29:53

SimpleSums (SimpleSum

Submission received
Your submission has been received and is currently being evaluated.

Score: 0 / 100

Submission details

#	Outcome	Details
1	Not correct	Output isn't correct

Compilation output

Compilation outcome:	Compilation succeeded
Compilation time:	0.000 sec
Memory used:	0 bytes

Standard output

Standard error

Close

Figure 10: How a wrong answer outcome is displayed

If you upload a correct solution, you will see something like:

TestContest
Logged in as Test User (Dummy)
Logout
Automatic

Server time:
21:30:53
Time left:
60290:29:06

[Overview](#)
[Communication](#)
SIMPLESUMS
[Statement](#)
[Submissions](#)
[Documentation](#)
[Testing](#)

Contest Management System
is released under the GNU
Affero General Public
License.

SimpleSums (SimpleSums) submissions

Score:
100 / 100

Submit a solution

SimpleSums: Choose File No file chosen
C++11 / g++
Submit Reset

Previous submissions

Time	Status		Score	Files
9:30:45 PM	Evaluated	details	100 / 100	Download
9:29:46 PM	Evaluated	details	0 / 100	Download

Figure 11: How a correct answer outcome is displayed

There are a few other possible outcomes you will see

- Execution timed out** To test whether solutions are *efficient* as well as correct, we only award points to a submission if it takes less than 1 second to solve each test case. For each test case where a solution was not returned in under 1 second, this verdict will be displayed.
- Memory limit exceeded** Similarly, if a solution used too much RAM when it is run, this verdict will be returned. This is rarely seen, unless your solution uses extreme quantities of memory (usually >500MB).
- Execution failed ...** Your program crashed, i.e. raised an exception, after encountering some bug in *your* code. Try to figure out what potential *edge-case* inputs could cause your code to crash.

More information on each of these can be found in the documentation panel, on the left hand side in the server.

Input/Ouput in Python

There are a few useful general techniques in Python that you can use to perform input and output functions:

By default, the `input()` function returns a string, so reading strings is quite straightforward

```
a_string = input()
```

If you have a sentence of 'words' separated by spaces on a single line, that we want to save into a list of individual words we can use the builtin `split()` function

```
words = input().split()
```

If the words are separated by something other than spaces, e.g. "-", like

```
a-weirdly-presented-sentence
```

We can use

```
words = input().split("-")
```

This works for multiple separator characters

```
even--weirder--presented--sentence
```

```
words = input().split("--")
```

For a single int or float, we can just use:

```
intvar = int(input())
floatvar = float(input())
```

Which reads in a string that contains some number, and converts it to a numeric type (int or float respectively).

If you want to read a fixed number of space separated integers on a single line, you can use code like:

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

This is effectively taking the list of strings that we see above, and using the `map()` function, to turn each 'word' in the list into an integer. Then, these are saved into three variables.

If the inputs are separated by something other than spaces, e.g.

```
12:16:34
```

You can use

12

```
a,b,c = map(int, input(":".split()))
```

Similarly to above.

If the inputs are floating point numbers:

12.07:16.54:34.086

The code becomes:

```
a,b,c = map(float, input(":".split()))
```

If we have a variable number of space separated inputs, that we need to read into a list, we can use:

```
numbers = list(map(int, input().split()))
```