Given a list of strings of bracket characters: {}(), the string of brackets is balanced under the following

conditions:

1. It is the empty string.

2. If strings a and b are balanced, then ab is balanced.

3. If string a is balanced, then (a) and {a} are balanced.

Write a class that determines whether the brackets in each string are balanced and returns true if the string

is balanced, or false if it is not.

Example 0

s = [ "{}()", "{()}", "({()})" ]

s[0] exhibits condition 2 above. "{}" and "()" are balanced, so "{}()" is balanced. Return true.

s[1] exhibits condition 3 above. "()" is balanced, so "{()}" is balanced. Return true.

s[2] exhibits condition 3 above. "()" is balanced, so "{()}" is balanced and "({()})" is balanced. Return true.

Example 1

s = ["{}(", "({)}", "((", "}{" ]

s[0] → '"{}(" is an unbalanced string due to the open "(". Return false.

s[1] → "({)}" is an unbalanced string due to ")" before "{" has been closed. Return false.

s[2] → "((", is an unbalanced string because neither "(" is closed. Return false.

s[2] → "}{" is an unbalanced string because "}" comes before a "{" and because the final "{" is not closed.

Return false.

Function Description

The provided code contains the declaration for a class named Solution with a main method that does the

following:

Creates a Parser object.

Reads an unknown number of strings from stdin.

Passes each string as an argument to the Parser object's isBalanced method and prints value returned

by the method on a new line.

Complete the function an isBalanced

string s: a string of characters to check for balance

Returns :

bool : a boolean that denotes whether the string is balanced: true if the string is balanced, or false if it is

not

Constraints

Each string consists only of the characters {, }, (, and ).

Each string has fewer than 50 characters.

Input Format for Custom Testing

Input from stdin will be processed as follows and passed to your Parser.isBalanced method.

Each line contains a string to parse.

=========================================================================================================================================================

Question 2:

Java Regex: Validating IPv4 Addresses

Pv4 was the first publicly used Internet Protocol. It used 4-byte addresses and permitted 2 distinct

values. The typical format for an IPv4 address is A.B.C.D where A, B, C, and D are all integers in the

inclusive range between 0 and 255, with leading zeros optional. For example:

Valid IPv4 Addresses

000.12.12.034

121.234.12.12

23.45.12.56

0.1.2.3

Invalid IPv4 Addresses

000.12.234.23.23

666.666.23.23

.213.123.23.32

23.45.22.32.

I.Am.not.an.ip

Complete the code in the editor below by replacing the blank (i.e., "\_\_\_\_\_\_\_\_") with a regular expression

that matches the valid IPv4 address. The locked code in the editor prints Valid for each correct match and

Invalid for each incorrect match.

Constraints

1 ≤ query ≤ 100

Input Format For Custom Testing

The first line contains the value of query describing the total number of queries.

Each of the next query lines contains a string to test for validity.

Sample Case 0

STDIN Function

----- --------

6 → query = 6

000.12.12.034 → each of the following queries is read and passed

to the regex

121.234.12.12

23.45.12.56

00.12.123.123123.123

122.23

Hello.IP

Sample Output 0

Valid

Valid

Valid

Invalid

Invalid

Invalid

Explanation 0

Valid IPv4 addresses must contain 4 integer values in the range [0-255] separated by dots.

=======================================================================================================================================

Question 3:

Simple Customer Support Ticketing

Customer support for an e-commerce business uses a computerized algorithm for evaluating if support

tickets are still open. Open and closed tickets are represented by different open and closed braces

respectively. For example, each of the braces '( [ {' represent an open ticket and need matching braces '} ]

)' in that order to close them.

The braces in a string are balanced if the following conditions are met:

\* All open braces must be closed

\* Each closed brace must have a matching open brace

\* Any set of nested braces must be closed before its surrounding braces

Given a 2-dimensional array of strings comprised of braces, verify that the braces in each string are

balanced. Return 'YES' if the braces are balanced and 'NO' otherwise.

Example

braces = ['[{}]', '[{]}']

\* The braces in the first string '[{}]' are balanced, because all braces are closed and all nested braces are closed in order.

\* The braces in the second string '[{]}' are not balanced, because the nested brace '{' was not closed before its surrounding '[ ]', so the order was not respected.

\* The result is ['YES', 'NO']

Function Description

Complete the function matchingBraces in the editor below.

matchingBraces has the following parameter(s): string braces[n]: an array of strings to analyze

Returns:

string[]: an array of strings, eaither 'YES' or 'NO', where the string at each index i denotes whether the braces are balanced in braces[i].

Constraints

\* 1 ≤ n ≤ 15

\* 1 ≤ length of each braces[i] ≤ 100

\* Each braces[i] consists of (, ), {, }, [, and ] only.

Input Format For Custom Testing

Input from stdin will be processed as follows and passed to the function

The first line contains an integer n, the number of elements in braces. Each of the next n lines contains a string that describes braces[i] where 0 ≤ i < n.

Sample Case 0

Sample Input

STDIN Function

----- -----

2 → braces[] size n = 2

{}[]() → braces = ['{}[]()', '{[}]}']

{[}]}

Sample Output

YES

NO

Explanation

The braces in the first string '{}[]()' are balanced, because all braces are closed

The braces in the second string '{[}]}' are not balanced, because the nested braces '{[' were not

closed in order '}]' and not all open and closed braces match

The result is ['YES', 'NO']

=======================================================================================================================================

QUESTION 4

The Adder Class

There are different kinds of calculators which are available in the market for different purposes. Sam wants

to make a calculator which can return the sum of two integers.

Implement the Adder class which should follow the following:

It should inherit from the Calculator class .

It should implement the method add(int a, int b) which should calculate and return the sum

of two integer parameters, a and b.

The locked stub code in the editor consists of the following:

An abstract class named Calculator which contains an abstract method, add(int a, int b) .

A solution class which

creates an object of the Adder class.

reads the inputs and passes them in a method called by the object of the Adder class.

Constraints

0 < a, b < 105

Input Format For Custom Testing

The only line contains two space-separated integers, a and b.

Sample Case 0

Sample Input For Custom Testing

1 1

Sample Output

The sum is: 2

Explanation

When the add method is called with the arguments a = 1 and b = 1, it calculates and returns their sum

as 1 + 1 = 2, which is then printed.

Sample Case 1

Sample Input For Custom Testing

2 3

Sample Output

5

Explanation

When the add method is called with the arguments a = 2 and b = 3, it calculates and returns their sum

as 2 + 3 = 5, which is then printed.

=============================================================================================================================

QUESTION 5

Java: Find the Output Multiple Choice

Consider the following Java code snippet:

public int divide(int a, int b) {

int c = -1;

try {

c = a / b;

}

catch (Exception e) {

System.err.print("Exception ");

}

finally {

System.err.println("Finally ");

}

return c;

}

What will our code print when we call divide(4, 0)?

Options:

**Exception Finally - Ans**

Finally Exception

Exception

Finally

No output

-1

====================================================================================================================

QUESTION 6 -> **1The Class objects are constructed by the JVM as classes are loaded by an instance of java.lang.ClassLoader**

Java : Object Class Multiple Choice

Which of the following is true?

Options:

The Class class is the superclass of the Object class.

The Object class is final.

The Class objects are constructed by the JVM as classes are loaded by an instance of java.lang.ClassLoader

None of the above

=============================================================================================================

QUESTION 7 -> Ans: **String temp [] = {"a", "b", "c"};**

Java: String Array Multiple Choice

Which of the following Java declaration of the String array is correct?

Options:

String temp [] = new String {"j" "a" "z"};

String temp [] = { "j " " b" "c"};

String temp = {"a", "b", "c"};

String temp [] = {"a", "b", "c"};

==============================================================================================

QUESTION 8

Java: Inner Classes Multiple Choice -- **Ans -> Sample**

Examine the following Java code, it includes an inner class, what will be the output of the following code?

public final class Test4 {

class Inner {

void test() {

if (Test4.this.flag) {

sample();

}

}

}

private boolean flag = true;

public void sample() {

System.out.println("Sample");

}

public Test4() {

(new Inner()).test();

}

public static void main(String args []) {

new Test4();

}

}

Options:

Prints out "Sample"

Program produces no output but terminates correctly

Program does not terminate.

The program will not compile

File uploads and downloads are integral parts of many web applications. In this problem, you will be working with a project that provides a REST API for file upload and download operations with a few validations.

The definitions and detailed requirements list follow. You will be graded on whether your application perfor ms data retrieval and manipulation based on given use cases exactly as described in the requirements.

Validations to be performed:

If a file size exceeds 100KB, return the status code INTERNAL\_SERVER\_ERROR and don't store the file in the database.

You are expected to use a file size limit constraint using configuration instead of doing it programmatically.

The REST service needs to expose 2 API endpoints for file uploads and downloads. All the uploaded files need to be stored in the local file system, which needs to be at UPLOAD\_DIR=PROJECT\_ROOT/uploads

POST request to /uploader :

Receives two parameters, fileName and file

stores the file in UPLOAD\_DIR and returns status code 201 as a response

if the user uploads the same fileName again, the previous file should be replaced with the latest one and return status code 201

GET request to /downloader :

accepts fileName as a request parameter

if the file exists, it should return the file with status code 200

if the file doesn't exist, it should return status code 404

Your task is to complete the given project so that it passes all the test cases when running the provided unit tests

Example requests and responses

POST request to /uploader

Request body:

{ "fileName": "test\_file.txt", "file": content of test\_file.txt }

The response code is 201. This puts the file into the UPLOAD\_DIR.

GET request to /downloader

Request body:

{ "fileName": "test\_file.txt" }

The response code is 200, and it returns test\_file.txt as the response.

GET request to /downloader

Request body:

{ "fileName": "test\_file2.txt" }

The response code is 404 because the file doesn't exist.