CodeForge - B01 - Nối Chuỗi Với concat()

Độ khó: ★ Easy

Đề bài

Nối hai chuỗi sử dụng method `concat()`.

Input

- Dòng 1: Chuỗi thứ nhất
- Dòng 2: Chuỗi thứ hai

Output

- In ra chuỗi sau khi nối

Constraints

- `0 ≤ độ dài mỗi chuỗi ≤ 10^5`

Ví dụ

Test case 1

Input:

```
Hello
World
```

Output:

```
HelloWorld
```

Test case 2

Input:

```
Java
Programming
```

Output:

```
JavaProgramming
```

Test case 3

Input:

```
Hello
 World
```

Output:

```
Hello  World
```

Tags: `string`, `concat`, `concatenation`

CodeForge - B02 - Thay Thế Ký Tự

Độ khó: ★ Easy

Đề bài

Thay thế tất cả các ký tự cũ bằng ký tự mới (sử dụng `replace(char, char)`).

Input

- Dòng 1: Chuỗi gốc
- Dòng 2: Ký tự cũ
- Dòng 3: Ký tự mới

Output

- In ra chuỗi sau khi thay thế

Constraints

- `1 ≤ độ dài chuỗi ≤ 10^5`

Ví dụ

Test case 1

Input:

```
Hello World
o
a
```

Output:

```
Hella Warld
```

Test case 2

Input:

```
Java Programming
a
e
```

Output:

```
Jeve Progremming
```

Test case 3

Input:

```
abcabc
a
x
```

Output:

```
xbcxbc
```

Tags: `string`, `replace`, `character`

CodeForge - B03 - Thay Thế Chuỗi Con

Độ khó: ★★ Medium

Đề bài

Thay thế tất cả các chuỗi con cũ bằng chuỗi mới (sử dụng `replace(CharSequence, CharSequence)`).

Input

- Dòng 1: Chuỗi gốc
- Dòng 2: Chuỗi con cũ
- Dòng 3: Chuỗi con mới

Output

- In ra chuỗi sau khi thay thế

Constraints

- `1 ≤ độ dài chuỗi gốc ≤ 10^5`
- `1 ≤ độ dài chuỗi con ≤ 1000`

Ví dụ

Test case 1

Input:

```
Hello World Hello
Hello
Hi
```

Output:

```
Hi World Hi
```

Test case 2

Input:

```
Java Java Java
Java
Python
```

Output:

```
Python Python Python
```

Test case 3

Input:

```
abcabcabc
abc
xyz
```

Output:

```
xyzxyzxyz
```

Tags: `string`, `replace`, `substring`

CodeForge - B04 - Thay Thế Lần Đầu

Độ khó: ★★ Medium

Đề bài

Thay thế lần xuất hiện đầu tiên của chuỗi con (sử dụng `replaceFirst(String, String)`).

Input

- Dòng 1: Chuỗi gốc
- Dòng 2: Chuỗi con cũ
- Dòng 3: Chuỗi con mới

Output

- In ra chuỗi sau khi thay thế lần đầu

Constraints

- `1 ≤ độ dài chuỗi gốc ≤ 10^5`
- `1 ≤ độ dài chuỗi con ≤ 1000`

Ví dụ

Test case 1

Input:

```
Hello World Hello
Hello
Hi
```

Output:

```
Hi World Hello
```

Test case 2

Input:

```
Java Java Java
Java
Python
```

Output:

```
Python Java Java
```

Test case 3

Input:

```
abcabcabc
abc
xyz
```

Output:

```
xyzabcabc
```

Tags: `string`, `replaceFirst`, `substring`

CodeForge - B05 - Tách Chuỗi Theo Dấu Phân Cách

Độ khó: ★★ Medium

Đề bài

Tách chuỗi thành các phần theo dấu phân cách (sử dụng `split(String)`).

Input

- Dòng 1: Chuỗi gốc
- Dòng 2: Dấu phân cách

Output

- In ra các phần sau khi tách, mỗi phần trên một dòng

Constraints

- `1 ≤ độ dài chuỗi ≤ 10^5`

Ví dụ

Test case 1

Input:

```
apple,banana,orange
,
```

Output:

```
apple
banana
orange
```

Test case 2

Input:

```
Java-Python-C++
-
```

Output:

```
Java
Python
C++
```

Test case 3

Input:

```
one two three four
```

Output:

```
one
two
three
four
```

Tags: `string`, `split`, `tokenization`

CodeForge - B06 - Nối Mảng Chuỗi

Độ khó: ★★ Medium

Đề bài

Nối các phần tử trong mảng chuỗi thành một chuỗi với dấu phân cách (sử dụng `String.join()` - Java 8+).

Input

- Dòng 1: Số phần tử N
- Dòng 2: N chuỗi cách nhau bởi dấu cách
- Dòng 3: Dấu phân cách

Output

- In ra chuỗi sau khi join

Constraints

- `1 ≤ N ≤ 1000`
- `0 ≤ độ dài mỗi chuỗi ≤ 100`

Ví dụ

Test case 1

Input:

```
3
apple banana orange
,
```

Output:

```
apple,banana,orange
```

Test case 2

Input:

```
4
Java Python C++ Go
-
```

Output:

```
Java-Python-C++-Go
```

Test case 3

Input:

```
2
Hello World
```

Output:

```
Hello World
```

Tags: `string`, `join`, `delimiter`, `java8`

CodeForge - B07 - Lặp Chuỗi

Độ khó: ★ Easy

Đề bài

Lặp lại chuỗi N lần (sử dụng `repeat(int)` - Java 11+).

Input

- Dòng 1: Chuỗi gốc
- Dòng 2: Số lần lặp N

Output

- In ra chuỗi sau khi lặp

Constraints

- `1 ≤ độ dài chuỗi ≤ 1000`
- `0 ≤ N ≤ 100`

Ví dụ

Test case 1

Input:

```
Ha
3
```

Output:

```
HaHaHa
```

Test case 2

Input:

```
Java
2
```

Output:

```
JavaJava
```

Test case 3

Input:

```
Hello
0
```

Output:

```

```

Tags: `string`, `repeat`, `java11`

CodeForge - B08 - Định Dạng Chuỗi

Độ khó: ★★ Medium

Đề bài

Định dạng chuỗi theo pattern (sử dụng `String.format()`).

Pattern: "Name: %s, Age: %d, Score: %.2f"

Input

- Dòng 1: Tên (String)
- Dòng 2: Tuổi (int)
- Dòng 3: Điểm (double)

Output

- In ra chuỗi đã định dạng

Constraints

- `1 ≤ độ dài tên ≤ 100`
- `0 ≤ tuổi ≤ 150`
- `0.0 ≤ điểm ≤ 100.0`

Ví dụ

Test case 1

Input:

```
Alice
20
95.5
```

Output:

```
Name: Alice, Age: 20, Score: 95.50
```

Test case 2

Input:

```
Bob
25
88.75
```

Output:

```
Name: Bob, Age: 25, Score: 88.75
```

Test case 3

Input:

```
Charlie
30
100.0
```

Output:

```
Name: Charlie, Age: 30, Score: 100.00
```

Tags: `string`, `format`, `formatting`

CodeForge - B09 - StringBuilder Basics

Độ khó: ★ Easy

Đề bài

Tạo StringBuilder, append các chuỗi và in ra kết quả.

◇ Input

- Dòng 1: Số lượng chuỗi N
- N dòng tiếp theo: Mỗi dòng một chuỗi

◇ Output

- In ra chuỗi sau khi append tất cả

◇ Constraints

- `1 ≤ N ≤ 1000`
- `0 ≤ độ dài mỗi chuỗi ≤ 100`

Ví dụ

Test case 1

Input:

```
3
Hello
World
Java
```

Output:

```
HelloWorldJava
```

Test case 2

Input:

```
2
Good
Morning
```

Output:

```
GoodMorning
```

Test case 3

Input:

```
1
Single
```

Output:

```
Single
```

Tags: `stringbuilder`, `append`, `mutable`

CodeForge - B10 - StringBuilder Insert

Độ khó: ★★ Medium

Đề bài

Sử dụng `insert()` để chèn chuỗi vào vị trí chỉ định.

Input

- Dòng 1: Chuỗi gốc
- Dòng 2: Vị trí chèn (0-indexed)
- Dòng 3: Chuỗi cần chèn

Output

- In ra chuỗi sau khi insert

Constraints

- `1 ≤ độ dài chuỗi gốc ≤ 10^5`
- `0 ≤ vị trí ≤ độ dài chuỗi`
- `1 ≤ độ dài chuỗi chèn ≤ 1000`

Ví dụ

Test case 1

Input:

```
HelloWorld
5
Java
```

Output:

```
HelloJavaWorld
```

Test case 2

Input:

```
abcdef
3
```

```
XYZ
```

Output:

```
abcXYZdef
```

Test case 3

Input:

```
Hello
0
Start
```

Output:

```
StartHello
```

Tags: `stringbuilder`, `insert`, `manipulation`

CodeForge - B11 - StringBuilder Delete

Độ khó: ★★ Medium

Đề bài

Sử dụng `delete()` để xóa một đoạn trong StringBuilder.

Input

- Dòng 1: Chuỗi gốc
- Dòng 2: Vị trí bắt đầu và kết thúc (start, end)

Output

- In ra chuỗi sau khi delete

Constraints

- `1 ≤ độ dài chuỗi ≤ 10^5`
- `0 ≤ start < end ≤ độ dài chuỗi`

Ví dụ

Test case 1

Input:

```
Hello World
5 11
```

Output:

```
Hello
```

Test case 2

Input:

```
abcdefgh
2 5
```

Output:

```
abfgh
```

Test case 3

Input:

```
JavaProgramming
4 15
```

Output:

```
Java
```

Tags: `stringbuilder`, `delete`, `manipulation`

CodeForge - B12 - StringBuilder Reverse

Độ khó: ★ Easy

Đề bài

Đảo ngược chuỗi sử dụng `StringBuilder.reverse()`.

Input

- Một dòng chứa chuỗi

Output

- In ra chuỗi sau khi đảo ngược

Constraints

- `1 ≤ độ dài chuỗi ≤ 10^5`

Ví dụ

Test case 1

Input:

```
Hello
```

Output:

```
olleH
```

Test case 2

Input:

```
Java Programming
```

Output:

```
gnimmargorP avaJ
```

Test case 3

Input:

```
12345
```

Output:

```
54321
```

Tags: `stringbuilder`, `reverse`, `utility`

CodeForge - B13 - StringBuilder Capacity vs Length

Độ khó: ★★ Medium

Đề bài

Hiểu sự khác biệt giữa `capacity()` và `length()` của StringBuilder.

Input

- Một dòng chứa chuỗi khởi tạo

Output

- Dòng 1: Length
- Dòng 2: Capacity

Constraints

- `0 ≤ độ dài chuỗi ≤ 10^5`

Ví dụ

Test case 1

Input:

```
Hello
```

Output:

```
5
21
```

Giải thích: Length = 5, Capacity = 16 + 5 = 21

Test case 2

Input:

```

```

Output:

```
0
16
```

Giải thích: Empty StringBuilder có capacity mặc định 16

Test case 3

Input:

```
Java
```

Output:

```
4
20
```

Tags: `stringbuilder`, `capacity`, `length`, `internals`

CodeForge - B14 - StringBuilder DeleteCharAt

Độ khó: ★ Easy

Đề bài

Xóa ký tự tại vị trí chỉ định (sử dụng `deleteCharAt()`).

Input

- Dòng 1: Chuỗi gốc
- Dòng 2: Vị trí cần xóa (0-indexed)

Output

- In ra chuỗi sau khi xóa

Constraints

- `1 ≤ độ dài chuỗi ≤ 10^5`
- `0 ≤ vị trí < độ dài chuỗi`

Ví dụ

Test case 1

Input:

```
Hello
1
```

Output:

```
Hllo
```

Test case 2

Input:

```
Java
0
```

Output:

```
ava
```

Test case 3

Input:

```
abcdef
5
```

Output:

```
abcde
```

Tags: `stringbuilder`, `deleteCharAt`, `manipulation`

CodeForge - B15 - StringBuilder SetCharAt

Độ khó: ★ Easy

Đề bài

Thay đổi ký tự tại vị trí chỉ định (sử dụng `setCharAt()`).

◇ Input

- Dòng 1: Chuỗi gốc
- Dòng 2: Vị trí cần thay đổi (0-indexed)
- Dòng 3: Ký tự mới

◇ Output

- In ra chuỗi sau khi thay đổi

◇ Constraints

- `1 ≤ độ dài chuỗi ≤ 10^5`
- `0 ≤ vị trí < độ dài chuỗi`

Ví dụ

Test case 1

Input:

```
Hello
0
J
```

Output:

```
Jello
```

Test case 2

Input:

```
Java
3
i
```

Output:

```
Javi
```

Test case 3

Input:

```
abcdef
2
X
```

Output:

```
abXdef
```

Tags: `stringbuilder`, `setCharAt`, `manipulation`

CodeForge - B16 - So Sánh String vs StringBuilder Performance

Độ khó: ★★ Medium

Đề bài

Nối N chuỗi sử dụng StringBuilder (đúng cách).

Input

- Dòng 1: Số lượng chuỗi N
- N dòng tiếp theo: Mỗi dòng một chuỗi

Output

- In ra chuỗi sau khi nối tất cả

Constraints

- `1 ≤ N ≤ 10^5`
- `1 ≤ độ dài mỗi chuỗi ≤ 100`

Ví dụ

Test case 1

Input:

```
5
Hello
World
From
Java
StringBuilder
```

Output:

```
HelloWorldFromJavaStringBuilder
```

Test case 2

Input:

```
3
a
b
c
```

Output:

```
abc
```

Test case 3

Input:

```
2
Good
Morning
```

Output:

```
GoodMorning
```

Tags: `stringbuilder`, `performance`, `concatenation`

CodeForge - B17 - Chuỗi Đối Xứng Với StringBuilder

Độ khó: ★★ Medium

Đề bài

Kiểm tra chuỗi có đối xứng không sử dụng StringBuilder.reverse().

Input

- Một dòng chứa chuỗi

Output

- In ra `YES` nếu đối xứng, `NO` nếu không

Constraints

- `1 ≤ độ dài chuỗi ≤ 10^5`

Ví dụ

Test case 1

Input:

```
racecar
```

Output:

```
YES
```

Test case 2

Input:

```
hello
```

Output:

```
NO
```

Test case 3

Input:

```
madam
```

Output:

```
YES
```

Tags: `stringbuilder`, `palindrome`, `reverse`

CodeForge - B18 - Xóa Khoảng Trắng Thừa

Độ khó: ★★ Medium

Đề bài

Xóa tất cả khoảng trắng thừa, chỉ giữ lại 1 khoảng trắng giữa các từ (sử dụng StringBuilder).

◇ Input

- Một dòng chứa chuỗi (có thể có nhiều khoảng trắng)

◇ Output

- In ra chuỗi đã chuẩn hóa

◇ Constraints

- `1 ≤ độ dài chuỗi ≤ 10^5`

Ví dụ

Test case 1

Input:

```
Hello    World    Java
```

Output:

```
Hello World Java
```

Test case 2

Input:

```
   Good   Morning
```

Output:

```
Good Morning
```

Test case 3

Input:

```
a  b  c  d
```

Output:

```
a b c d
```

Tags: `stringbuilder`, `whitespace`, `normalization`

CodeForge - B19 - StringBuilder ToString

Độ khó: ★ Easy

Đề bài

Xây dựng chuỗi với StringBuilder, sau đó convert sang String với `toString()`.

Input

- Dòng 1: Số lượng phần tử N
- N dòng tiếp theo: Các chuỗi

Output

- In ra chuỗi cuối cùng sau khi toString()

Constraints

- `1 ≤ N ≤ 1000`
- `1 ≤ độ dài mỗi chuỗi ≤ 100`

Ví dụ

Test case 1

Input:

```
3
Java
is
fun
```

Output:

```
Javaisfun
```

Test case 2

Input:

```
2
Hello
World
```

Output:

```
HelloWorld
```

Test case 3

Input:

```
1
Single
```

Output:

```
Single
```

Tags: `stringbuilder`, `toString`, `conversion`

CodeForge - B20A - Tạo Chuỗi Lớn Hiệu Quả

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Tạo một chuỗi bằng cách nối N chuỗi con, mỗi chuỗi lặp lại M lần.

Sử dụng StringBuilder để tối ưu performance.

Input

- Dòng 1: Số lượng chuỗi con N
- Dòng 2: Số lần lặp M
- N dòng tiếp theo: Các chuỗi con

Output

- In ra độ dài của chuỗi kết quả
- In ra 50 ký tự đầu tiên (nếu có)

Constraints

- `1 ≤ N ≤ 1000`
- `1 ≤ M ≤ 1000`
- `1 ≤ độ dài chuỗi con ≤ 100`

Ví dụ

Test case 1

Input:

```
3
2
Hello
World
Java
```

Output:

```
30
HelloWorldJavaHelloWorldJava
```

Test case 2

Input:

```
2
3
ab
cd
abcdabcdabcd
```

Output:

```
12
abcdabcdabcd
```

Tags: `stringbuilder`, `performance`, `large-string`

CodeForge - B21A - Đảo Ngược Từng Từ

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Đảo ngược từng từ trong chuỗi nhưng giữ nguyên thứ tự các từ.

Ví dụ: "Hello World" → "olleH dlroW"

Sử dụng StringBuilder.

Input

- Một dòng chứa chuỗi (các từ cách nhau bởi khoảng trắng)

Output

- In ra chuỗi sau khi đảo ngược từng từ

Constraints

- `1 ≤ độ dài chuỗi ≤ 10^5`

Ví dụ

Test case 1

Input:

```
Hello World
```

Output:

```
olleH dlroW
```

Test case 2

Input:

```
Java Programming Language
```

Output:

```
avaJ gnimmargorP egaugnaL
```

Test case 3

Input:

```
abc def ghi
```

Output:

```
cba fed ihg
```

Tags: `stringbuilder`, `reverse`, `word-processing`

CodeForge - B22A - Loại Bỏ Ký Tự Trùng Liên Tiếp

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Loại bỏ các ký tự trùng lặp liên tiếp, chỉ giữ lại 1 ký tự.

Ví dụ: "aaabbccca" → "abca"

Sử dụng StringBuilder.

Input

- Một dòng chứa chuỗi

Output

- In ra chuỗi sau khi loại bỏ trùng lặp liên tiếp

Constraints

- `1 ≤ độ dài chuỗi ≤ 10^5`

Ví dụ

Test case 1

Input:

```
aaabbccca
```

Output:

```
abca
```

Test case 2

Input:

```
aabbccdd
```

Output:

```
abcd
```

Test case 3

Input:

```
abcdef
```

Output:

```
abcdef
```

Test case 4

Input:

```
aaaa
```

Output:

```
a
```

Tags: `stringbuilder`, `deduplication`, `consecutive`

CodeForge - B23A - Xây Dựng CSV String

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Xây dựng chuỗi CSV từ nhiều dòng dữ liệu.

Format: "Name,Age,Score"

Sử dụng StringBuilder để tối ưu.

◇ Input

- Dòng 1: Số người N
- N nhóm 3 dòng tiếp theo:
  - Tên
  - Tuổi
  - Điểm

◇ Output

- In ra N dòng CSV

◇ Constraints

- `1 ≤ N ≤ 1000`
- `1 ≤ độ dài tên ≤ 50`
- `0 ≤ tuổi ≤ 150`
- `0.0 ≤ điểm ≤ 100.0`

Ví dụ

Test case 1

Input:

```
2
Alice
20
95.5
Bob
25
88.0
```

Output:

```
Alice,20,95.5
Bob,25,88.0
```

Test case 2

Input:

```
1
Charlie
30
100.0
```

Output:

```
Charlie,30,100.0
```

Tags: `stringbuilder`, `csv`, `formatting`

CodeForge - B24A - Chèn Dấu Phân Cách

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

Chèn dấu phân cách vào giữa mỗi ký tự trong chuỗi.

Ví dụ: "abc" với dấu "-" → "a-b-c"

Sử dụng StringBuilder.

Input

- Dòng 1: Chuỗi gốc
- Dòng 2: Dấu phân cách

Output

- In ra chuỗi sau khi chèn dấu phân cách

Constraints

- `1 ≤ độ dài chuỗi ≤ 10^5`
- `1 ≤ độ dài dấu phân cách ≤ 10`

Ví dụ

Test case 1

Input:

```
abc
-
```

Output:

```
a-b-c
```

Test case 2

Input:

```

```

```
Hello
*
```

Output:

```
H*e*l*l*o
```

Test case 3

Input:

```
Java
::
```

Output:

```
J::a::v::a
```

Test case 4

Input:

```
a
-
```

Output:

```
a
```

Tags: `stringbuilder`, `delimiter`, `insertion`

CodeForge - B25A - StringBuilder vs String Concatenation Benchmark

Độ khó: ★ ★ ★ Hard (Advanced)

Đề bài

So sánh hiệu suất giữa String concatenation và StringBuilder.

Nối N chuỗi bằng cả 2 cách, đo thời gian và so sánh.

Input

- Dòng 1: Số lượng chuỗi N
- N dòng tiếp theo: Các chuỗi

Output

- Dòng 1: Kết quả nối (chỉ 100 ký tự đầu)
- Dòng 2: Thời gian String concatenation (ms)
- Dòng 3: Thời gian StringBuilder (ms)
- Dòng 4: Kết luận (StringBuilder nhanh hơn bao nhiêu lần)

Constraints

- `100 ≤ N ≤ 10000`
- `1 ≤ độ dài mỗi chuỗi ≤ 50`

Ví dụ

Test case 1

Input:

```
1000
Hello
World
Java
...
(997 dòng nữa)
```

Output:

```
HelloWorldJava... (100 chars)
String: 245 ms
```

```
StringBuilder: 3 ms
StringBuilder is 81.67x faster
```

Test case 2

Input:

```
5000
a
b
c
...
(4997 dòng nữa)
```

Output:

```
abc... (100 chars)
String: 1523 ms
StringBuilder: 7 ms
StringBuilder is 217.57x faster
```

Tags: stringbuilder, performance, benchmark, optimization