题目描述

给你两个字符串if和p，要求从if中找到一个和p相同的连续子串，并输出该子串第一个字符的下标。

输入描述

- 输入文件包括两行 分别表示字符串if和p
- 保证if的长度不小于p
- if的长度不超过1000000
- p的长度不超过10000

输出描述

- 如果if从if中找到一个和p相等的连续子串，则输出该子串第一个字符在if中的下标，下标从左到右依次为1,2,3,...；
- 如果不能，则输出 "No"
- 如果含有多个这样的子串，则输出第一个字符下标最小的

用例

输入	AVERDXIVYERDIAN RDXI
输出	4
说明	无

解析说明

这题目的应该是想考察子串 查找算法🔍，比如BM暴力算法，BMP算法，以及Horspool算法。

但是高级语言大多已经内置实现了这些算法，比如Java的String的indexOf，JS的String.prototype.indexOf，Python的find函数

Java算法源码

```java
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println(getResult(sc.nextLine(), sc.nextLine()));
  }

  public static String getResult(String str, String subStr) {
    if (str.length() < subStr.length()) {
      return "No";
    }

    int idx = str.indexOf(subStr);
    if (idx == -1) return "No";
    else return idx + 1 + "";
  }
}
```

JS算法代码

```javascript
// JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
rl.on("line", (line) => {
  lines.push(line);
  if (lines.length === 2) {
    getSubIndex(...lines);
    lines.length = 0;
  }
});

/* 寻找if子串 */
function getSubIndex(str, substr) {
  if (str.length < substr.length) {
    return console.log("No");
  }

  const index = str.indexOf(substr);
  if (index === -1) {
    return console.log("No");
  } else {
    return console.log(index + 1);
  }
}
```

Python算法源码

```python
# 输入获取
str = input()
subStr = input()


# 算法入口
def getResult():
    if len(str) < len(subStr):
        return "No"

    idx = str.find(subStr)
    if idx == -1:
        return "No"
    else:
        return idx + 1


# 算法调用
print(getResult())
```