ปัญหา ข้อความที่สนใจอยู่หนใดบ้าง (indexOf2)

จงเขียนโปรแกรมที่รับอินพุตมาเป็นข้อความ 2 บรรทัด เช่น

```
silpakorn university
si
```

จากนั้นโปรแกรมจะค้นหาว่าข้อความในบรรทัดที่สอง อยู่ตำแหน่งใดบ้างในข้อความแรก เช่น ในตัวอย่างข้างบนจะแสดงเลข **1** และ **17** ออกมาเป็นคำตอบ ในกรณีที่ไม่พบข้อความที่สองปรากฏในข้อความแรก ให้พิมพ์ว่า `string not found`

ตัวอย่าง

ข้อมูลเข้า	**ผลลัพธ์**
`silpakorn university` `si`	`1 17`
`silpakorn university` `paka`	`string not found`
`baaaa abaaabaaa` `aaa`	`2 3 9 13`

ข้อสังเกต ในตัวอย่างที่สาม ข้อความที่สนใจสองตำแหน่งมีการซ้อนเหลื่อมกันได้ในข้อความแรก (ตำแหน่งที่ 2 และ 3) เราต้องแสดงตำแหน่งออกมาให้ได้ทั้งหมด

คำแนะนำ เมธอด `indexOf` [ลิงค์] ที่ควรใช้ในข้อนี้ มีชื่อเหมือนกับเมธอด `indexOf` ที่ใช้ในข้อที่ผ่านมา **แต่รับพารามิเตอร์ไม่เหมือนกัน** โดยในตัวที่แนะนำให้ใช้ในข้อนี้จะยอมให้เราบอกด้วยว่า เราอยากจะค้นหานับจากตำแหน่งใด (หมายเลขตำแหน่งในภาษาจาวาจะเริ่มจาก 0 แต่ตำแหน่งในคำตอบของเราเริ่มจาก 1 ดังนั้นให้ระวังด้วย) เช่น หากเราเขียนว่า `int k = "baaa aaaaa".indexOf("aaa", 2);`

ค่าของ k ที่จได้จะเป็น 5 เพราะจะตรงกับอินเด็กซ์ 5 `"baaa aaaaa"` ตามที่ขีดเส้นใต้ไว้ สาเหตุที่ไม่เจอ `aaa` ตรงอินเด็กซ์ 1 ก็เพราะเราบอกให้ `indexOf` ค้นหาข้อความที่สนใจจากอินเด็กซ์ 2 เป็นต้นไปนั่นเอง

หมายเหตุ ชื่อเมธอดในภาษาจาวาจะซ้ำกันได้ หากมีพารามิเตอร์ที่ต่างกัน ไม่ว่าจะเป็นชนิดข้อมูลหรือตำแหน่ง ซึ่งหากความต่างนี้ทำให้สามารถแยกได้แบบเจาะจงว่าเราจะเรียกเมธอดใดกันแน่ จาวาจะยอมให้เราใช้ชื่อเมธอดเดิมซ้ำได้ เช่นใน

กรณีของ IndexOf ที่เราใช้ในข้อที่ผ่านมากับที่เราใช้ในข้อนี้ ได้มีการประกาศไว้ตามข้างล่างนี้

แบบข้อที่ผ่านมา `public int indexOf(String str)`

แบบในข้อนี้ `public int indexOf(String str, int fromIndex)`

สังเกตได้ว่า ถ้าหากเราเรียกใช้ `indexOf` เหมือนกัน แต่เราใส่พารามิเตอร์ตัวที่สองเพิ่มขึ้นไปอีกตัวเป็น `int` มันก็เป็นที่แน่ชัดว่าเราต้องการเรียกใช้ `indexOf` แบบที่ใช้ในข้อนี้ แต่ถ้าเราใส่พารามิเตอร์ไปเป็นสตริงแค่ตัวเดียว มันก็เป็นที่แน่ชัดเช่นกันว่าเราต้องการเรียกใช้ `indexOf` แบบข้อที่ผ่านมา

เทคนิคที่จาวายอมให้เมธอดชื่อซ้ำกันแต่พารามิเตอร์ต่างกันนี้เรียกว่า *การรับภาระเกิน* (*`Overloading`*) ซึ่งในที่นี้ก็คือการที่ชื่อเมธอดชื่อหนึ่งมีภาระหน้าที่มากกว่าหนึ่งอย่าง

ส่วนชุดพารามิเตอร์ที่ทำให้แยกได้ว่าเมธอดใดเป็นเมธอดใดนั้น มันแสดงถึงความเป็นเอกลักษณ์ของเมธอดนั้น เราเรียกมันว่า *ลายเซ็น* (*`signature`*) ในฐานะผู้เรียกใช้งาน เราจะต้องเข้าใจเรื่องลายเซ็นของเมธอดให้ดี ไม่อย่างนั้นเราอาจจะเรียกเมธอดผิดตัวก็เป็นได้

การรับภาระเกินเป็นเทคนิคที่ไม่มีในภาษาซี แต่มีในภาษา Java, C++ และภาษาอื่น ๆ อีกจำนวนมาก แต่ก็ไม่ใช่ว่าภาษาสมัยใหม่ทุกภาษาจะยอมให้ทำแบบนี้ได้ เพราะการใช้ชื่อเดิมซ้ำ ในบางแง่ก็สามารถสร้างความสับสนหรือกำกวมให้กับผู้ใช้งานหรือคอมพิวเตอร์ได้เช่นกัน ภาษาสมัยใหม่บางภาษา เช่น ruby จึงไม่ยอมให้มีการรับภาระเกินของเมธอด