3_Code_Guardian

(15分)
時間限制: 1 second
記憶體限制: 256 MB

題目敘述

自從 OpenAI 推出 ChatGPT，人工智能領域在過去的半年內快速演進。許多工作已開始透過與AI協作來更高效地滿足需求。其中，與電腦的溝通成為協作的瓶頸。因此，利用特定領域語言 (DSL, domain specific language) 來加速提問 (prompt) 成為關鍵之一。例如，在 Midjourney 上使用美術專用語句來產生圖片（例如：請使用「印象派畫法」來...），或者在正在研發中的 Mojo 語言中，使用 autotune 可以根據處理器核心數自動進行執行優化。

基於這個想法，再加上「懶」是人類進步的原動力，你需要寫一個程式，利用人工智能來幫助自己進行函式註解和錯誤處理，以便未來撰寫的程式更易於閱讀且能夠良好處理錯誤。為了方便說明與處理，假設程式碼的每一層縮排都是2個空白，空行將被替換為 "emptyline"，需要錯誤處理的地方將被標記為 "safe"。而你的程式將對它們進行以下處理：

- 將 "safe" 轉換為 "!!!(Analyze(<同一層未被解析的程式碼，去掉前後空白並以逗號隔開>))"。註：原本意思為 "ErrorCheck"，為了方便說明改為 "!!!"。
- 將 "emptyline" 轉換為 "//(Describe(程式碼區塊的起始行數，結束行數))"。註：原本意思為 "FunctionComment"，為了方便說明並節省打字，改為 "//"。為了簡化問題，只有前端沒有空白時才替換 emptyline, 當然 function 後的 emptyline 不屬於 function。
- 若縮排不正確，則僅輸出 "ERR!"。
- 最後一行為 "end" 時輸入的程式結束

輸入的程式碼	輸出的程式碼	以實際程式取代 ln# 較容易理解

emptyline
ln2
ln3
spsp ln4
spsp ln5
spspspsp ln6
spspspsp ln7
spspspsp ln8
spspspsp safe
spspspsp ln10
spspspsp safe
spspspsp ln12
spsp ln13
spsp ln14
spsp safe
ln16
emptyline
emptyline
ln19
spsp ln20
safe
end
	//Describe(2,16)
	ln2
	ln3
	spsp ln4
	spsp ln5
	spspspsp ln6
	spspspsp ln7
	spspspsp ln8
	spspspsp !!!Analyze(ln6,ln7,ln8)
	spspspsp ln10
	spspspsp !!!Analyze(ln10)
	spspspsp ln12
	spsp ln13
	spsp ln14
	spsp !!!Analyze(ln4,ln5,ln13,ln14)
	ln16
	emptyline
	//Describe(19,21)
	ln19
	spsp ln20
	!!!Analyze(ln19)
		# main - create markdown output
		@timer
		def main():
		  result = []
		  for line in sys.stdin.readlines():
		  line = line.rstrip()
		  line = line.replce(' ','ⓔ')
		  line = line+'
		'
		  raise if len(line) >= MAX_LINE_SIZE
		  result.append(line)
		  raise if len(result) >= MAX_ARRAY_SIZE
		  print('.',end='')
		  fh = open('missclose.txt','w+')
		  fh.write(''.join(line))
		  raise if filesize('missclose.txt') == 0
		# end of main
		
		# Entry Point
		if name == 'main':
		  main()
		raise if diskfull()

- 第 1個 emptyline 轉成需要 AI describe code 2-16 行，第2個 emptyline 沒有轉換因為沒有 function, 第3 個則轉19-21行。

- 第1個 safe 轉成需要 AI analyze “ln6,ln7,ln8” 的內容，第 2 個 safe 則只需分析 ln10，第 3 個 safe 需分析同層未分析的內容 (ln4,ln5,ln13,ln14), 以及最後一行的 safe 則只需分析第二個 function ln19 的內容。

PS: '␣' is ' ' (ASCII: 0x20) 字元

輸入格式

多行的文字，以 "end" 作為結束

輸出格式

輸出 “ERR!” 如果縮排錯誤
如無錯誤則輸出替換好的內容

資料範圍

英文字母、數字、空白、括弧、底線及換行字元

測試範例

輸入範例 1

```
emptyline
ln2
ln3
  ln4
  ln5
    ln6
    ln7
    ln8
    safe
    ln10
    safe
    ln12
  ln13
  ln14
  safe
ln16
emptyline
emptyline
ln19
  ln20
safe
end
```

輸出範例 1

```
//Describe(2,16)
ln2
ln3
  ln4
  ln5
    ln6
    ln7
    ln8
    !!!Analyze(ln6,ln7,ln8)
    ln10
    !!!Analyze(ln10)
    ln12
  ln13
  ln14
  !!!Analyze(ln4,ln5,ln13,ln14)
ln16
emptyline
//Describe(19,21)
ln19
  ln20
!!!Analyze(ln19)
```

輸入範例 2

```
emptyline
emptyline
ln3
emtpyline_safe_emptyline
  ln5
  safe
    ln7
    safe
      ln9
        ln10
          ln11
      safe
emptyline
ln14
end
```

輸出範例 2

```
emptyline
//Describe(3,12)
ln3
emtpyline_safe_emptyline
  ln5
  !!!Analyze(ln5)
    ln7
    !!!Analyze(ln7)
      ln9
        ln10
          ln11
      !!!Analyze(ln9)
//Describe(14,14)
ln14
```

輸入範例 3

```
emptyline
emptyline
emptyline
  safe
      safe
end
```

輸出範例 3

```
ERR!
```

範例說明

1-4行是原本應輸出如下，但因第5行內縮多一層所以只輸出 ERR!

```
emptyline
emptyline
//Describe(4,4)
  !!!Analyze()
```