

프로젝트과제 2

◆ 길동의 로그 파일

- ◆ 갑작스런 정전으로 인하여 편집하던 문서를 잃어버려, 길동은 컴퓨터에 남아있는 로그(log) 파일에서 문서를 복구하고자 한다.

◆ 로그 파일 형식 예: `e<<LEv<-o>>>`

이 형식은 실습을 위한 가상적인 형식 임.

1. 영문대소문자, 숫자, <, >, - 로 구성되며, 다수의 line으로 구성되어 있다 (줄당 최대 10,000 글자).
2. < : 키보드의 좌측 화살표 (←)를 사용하여 좌측으로 커서 (cursor)를 한 글자 이동한 것을 의미.
3. > : 우측으로 한 글자 커서를 이동한 것을 의미 (→).
4. - : 커서 좌측 글자를 하나 지운 것을 의미 (backspace).
5. 커서가 줄의 맨 좌측 (우측)에 있으면, < (>)는 무시.
6. 커서가 맨 좌측에 있으면, 지울 글자가 없어 - 는 무시.

◆ 문자열 복원 예(*).

로그 문자열

e<<LEv<-o>>>

로그 문자	화면 (는 커서)	의미 (키보드)
e	e	e 입력
<	e	←
<	e	← (커서 영향 없음)
L	L e	L 입력
E	L E e	E 입력
v	L E v e	v 입력
<	L E v e	←
-	L v e	E 지움 (backspace)
o	L o v e	o 입력
>	L o v e	→
>	L o v e	→
>	L o v e	→ (커서 영향 없음)

(*) 메모장으로 시뮬레이션 해보면 더욱 쉽게 이해할 수 있다.

◆ 추상화

다음과 같은 이름의 변수를 정의하자

string : 입력된 문자열 (로그 파일로부터)

deCoded : 현재 유지중인 글자들을 순서대로 보관하는 리스트.

pos : 현재 커서의 위치 (deCoded 리스트의 index).

◆ 패턴 인식

e<<LEv<-o>>>

◆ 앞에서 보인 예를 정의한 변수로 시뮬레이션 해보자.

시작

deCoded = [] pos = 0

e

deCoded = [e] pos = 1

<

deCoded = [e] pos = 0

<

deCoded = [e] pos = 0 커서위치 불변

e<<LEv<-o>>>

L deCoded = [L e] pos = 1

E deCoded = [L E e] pos = 2

v deCoded = [L E v e] pos = 3

< deCoded = [L E v e] pos = 2

- deCoded = [L v e] pos = 1

o deCoded = [L o v e] pos = 2

> deCoded = [L o v e] pos = 3

> deCoded = [L o v e] pos = 4

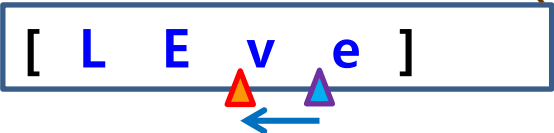
> deCoded = [L o v e] pos = 4 커서위치 불변

◆ 각 문자 별 처리

◆ '<' 인 경우

```
if pos > 0 : # 커서를 한 글자 좌측으로 이동 가능하면,  
    pos -= 1 # pos 값을 하나 줄인다.  
    # pos 값이 0이면 커서가 맨 왼쪽에 있으므로 그대로.
```

0 1 2 3 4=len(deCoded)
deCoded = [L E v e] pos = 3 → 2

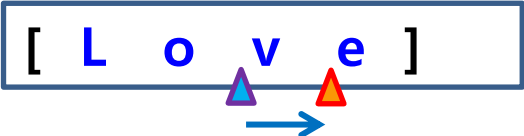


The diagram shows a list 'deCoded' with elements 'L', 'E', 'v', 'e' at indices 0, 1, 2, 3 respectively. The length of the list is 4. A red triangle at index 3 and a blue triangle at index 2 represent the cursor. A blue arrow points from the red triangle to the blue triangle, indicating a leftward movement from index 3 to index 2.

◆ '>' 인 경우

```
if pos < len(deCoded) : # 한 글자 우측으로 이동 가능하면,  
    pos += 1 # pos 값을 하나 늘린다.  
    # pos 값이 len(deCoded) 이면 그대로.
```

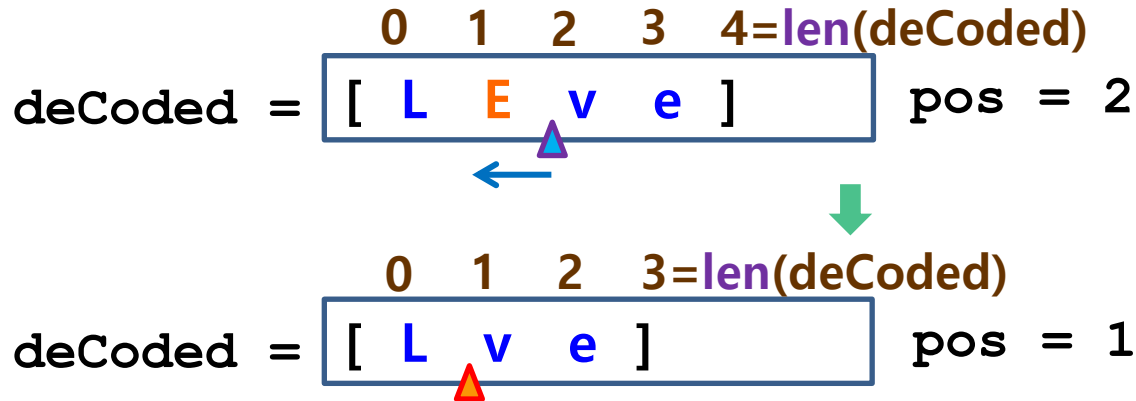
0 1 2 3 4=len(deCoded)
deCoded = [L o v e] pos = 2 → 3



The diagram shows a list 'deCoded' with elements 'L', 'o', 'v', 'e' at indices 0, 1, 2, 3 respectively. The length of the list is 4. A blue triangle at index 2 and a red triangle at index 3 represent the cursor. A blue arrow points from the blue triangle to the red triangle, indicating a rightward movement from index 2 to index 3.

◆ ' _ ' 인 경우

- ◆ pos != 0인 경우, 커서의 좌측에 지울 문자가 있다.



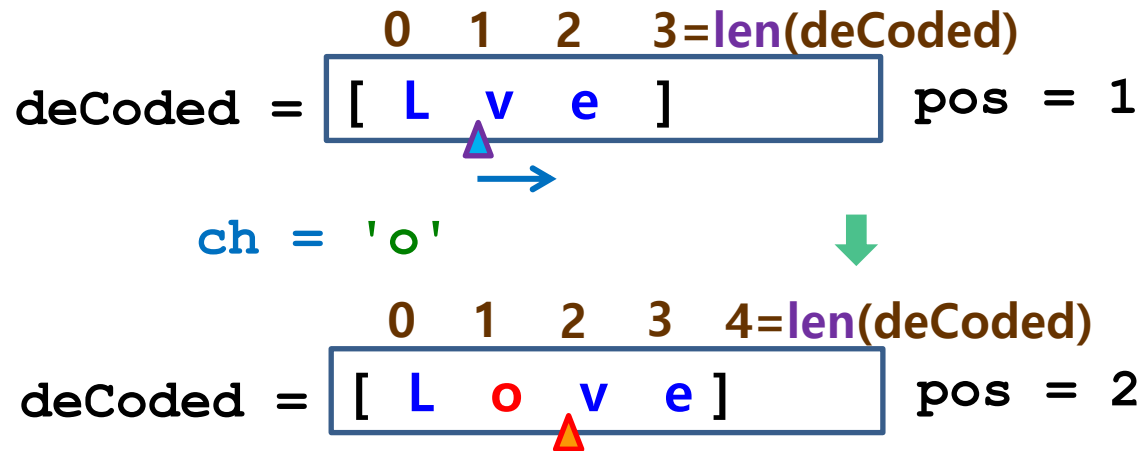
문자 'E'의 index가 pos 값보다 하나 작으므로 pos 값을 하나 줄인 후 `del` statement로 삭제한다.

- ◆ pos == 0이면, 커서의 좌측에 문자가 없으므로 무시한다.
- ◆ 따라서, 다음과 같이 처리한다.

```
if pos != 0 : # 지울 문자가 있다.  
    pos -= 1    # pos 값을 하나 줄인다.  
del deCoded[pos]
```

◆ 문자가 영문자이거나 숫자인 경우

- ◆ 이 경우에는 pos 값의 index에 문자를 끼워 넣고, pos 값을 하나 증가한다.



- ◆ 문자를 끼워 넣는 것은 리스트 slicing으로 해결한다.
위의 예에서는 `deCoded[pos:pos] = ch`
- ◆ 문자가 영문자 또는 숫자인지 판정은 string method를 사용하며, 만일 아닐 경우(' \n' 등) 무시한다.

```
ch.isalnum() # ch가 영문자이거나 숫자이면 True
```

◆ 이러한 문자별 처리 과정을 모아 복원 함수를 작성한다.

```
def decodeString(string) :  
    """ Recover a string from a keylog string  
        입력: string(문자열), 출력: 복원된 문자열 반환  
        deCoded[ ] : 복원된 문자열 리스트  
        pos : 현재 커서의 위치 (deCoded에서의 index)  
    """  
    deCoded = []  
    pos = 0  
    for ch in string : # string의 각 문자에 대해  
        pass           # 6,7,8 쪽의 경우에 따른 처리  
    return ''.join(deCoded) # .join()을 사용하여 문자열로 변환
```


◆ 실행 시간 측정을 포함한 전체 프로그램 형태

```
def decodeString(string) : # 복원 함수
    pass

# 메인 프로그램 시작
pass # 입력 파일과 출력 파일을 연다.

print("복원 시작")
pass #      입력 파일을 한 줄씩 읽어
      #      함수 decodeString을 호출하여 복원
      #      출력 파일에 복원된 문자열을 쓴다

print("복원 완료")
pass # 입력 파일과 출력 파일을 닫는다
```

파일
복원

◆ 입력 파일 및 프로그램 테스트

◆ 다운 받은 두 개의 파일은 다음과 같다.

`keylog_in.txt` : 입력 파일

`keylog_ans.txt` : `keylog_in_small.txt`의 복원 결과

◆ 프로그램을 돌려서 나온 출력파일과 `keylog_ans.txt`을 비교해본다.