

PL Practice Report



[03] Tail call split, combine

2021/03/22

201702083 최현석

▶ 주어진 문제

- Split 과 Combine 을 Tail - recursive 하게 구현하라

Split : Pair 를 원소로 갖는 List 를 인자로 받아

fst_Pair_List, snd_Pair_List 의 Pair 로 반환

Combine : 원소의 개수가 같은 두 List 를 인자로 받아 각 List 의 순서쌍들을 Pair 로
갖는 List 반환

Tail - recursive : 재귀 호출을 한 후 더 이상 처리할 게 없는 방식

▶ 해결 과정 - split

1. 입력 받은 List of Pair 를 split 후 결과 값으로 받기 위해 let - in 구문으로 List
두 개를 인자로 받는다.

```
↳ module F = Format
let split lst =

    let rec split_result lst l1 l2 =
```

2. Lst 의 상태에 따라 case 를 나눈다.

- Pair 존재 -> 재귀로 호출할 때 l1 과 l2 에 현 호출에서 추출한
Pair 의 fst, snd 원소를 :: 연산을 통해 추가해준다.
- Empty -> l1 과 l2 를 Pair 꼴로 반환한다.
이 때 :: 연산으로 추가되어 순서가 반대이므로
List.rev 를 사용하여 다시 정렬한다.

3. Let - in 이 끝나면 l1 과 l2 를 비어있는 리스트 [] 로 넣고 호출한다.

```
↳ split_result lst [] []
```

▶ 결과

```
macbook@MacDevCHS hw (main) $ ./_build/default/tailOpt.exe
Split this ! : [ ( 1 , 3 ) ( 3 , 5 ) ( 1 , 2 ) ( 4 , 5 ) ]
first_split : [ 1 3 1 4 ]
second_split : [ 3 5 2 5 ]
```

해결 과정 - combine

1. 입력 받은 두 List 를 combine 후 결과 값으로 받기 위해 let - in 구문으로 List 하나를 인자로 받는다.

```
let combine l1 l2 =  
  ↳  
    let rec combine_result l1 l2 acc =  
      (let _ = Printf.printf "%d\n" (List.length l1)) in
```

2. l1 과 l2 의 상태에 따라 case 를 나눈다.

- 둘다 Empty -> 결과 리스트인 acc 를 List.rev 를 통과시키고 반환한다.
- 둘 다 원소 존재 -> l1 과 l2 의 원소를 하나씩 추출하면 두 개의 tail list 가 된다.
다음 재귀 호출에서 인자로 tl1 tl2 (p1,p2)::acc 를 넣어 재귀 호출한다.

3. Let - in 이 끝나면 acc 를 비어있는 리스트 [] 로 넣고 호출한다.

```
↳ combine_result l1 l2 []
```

결과

```
A : [ 2 4 8 3 1 0 ]  
B : [ 100 150 123 1000 25 31 ]  
  
Combine A and B ! : [ ( 2 , 100 ) ( 4 , 150 ) ( 8 , 123 ) ( 3 , 1000 ) ( 1 , 25 ) ( 0 , 31 ) ]
```