HW6. Fib 문제

2017156019 영어교육
과제는 전적으로 혼자 했음 (Alone)

POA 2.6.5. function fib(n)

1. Comment Return F_n, the nth Fibonacci number
2. if n=0 then return(0) else # n이 0이면 0을 리턴한다.
3. last:=0, current:=1 # last의 초기값:0, current의 초기값:1
4. for i:=2 to n do # 2부터 n까지 i를 증가시키며 반복
5. temp:=last+current; last:=current; current:=temp
   # temp=last+current 임.
   # $last_{n+1} = current_n$ # current=temp
   ∴ $current_{n+1} = last_n + current_n$
6. return(current)

i) n=2, fib(2)

3. last:=0, current:=1
4. for i:=2 to 2 do
5. temp:=0+1=1; last:=1; current:=1
6. return(1)

⇒ temp=1, last=1, current=1

∴ fib(2)=1 //

ii) n=3, fib(3)

3. last:=0, current:=1
4. for i:=2 to 3 do
5. (i=2) temp:=0+1=1; last:=1; current:=1
   (i=3) temp:=1+1=2; last:=1; current:=2
6. return(2)

⇒ temp=2, last=1, current=2

∴ fib(3)=2

iii) n=7, fib(7)

3. last:=0, current:=1
4. for i:=2 to 7 do
5. (i=2) temp:=0+1=1; last:=1; current:=1
   (i=3) temp:=1+1=2; last:=1; current:=2
   (i=4) temp:=1+2=3; last:=2; current:=3
   (i=5) temp:=2+3=5; last:=3; current:=5
   (i=6) temp:=3+5=8; last:=5; current:=8
   (i=7) temp:=5+8=13; last:=8; current:=13
6. return(13)

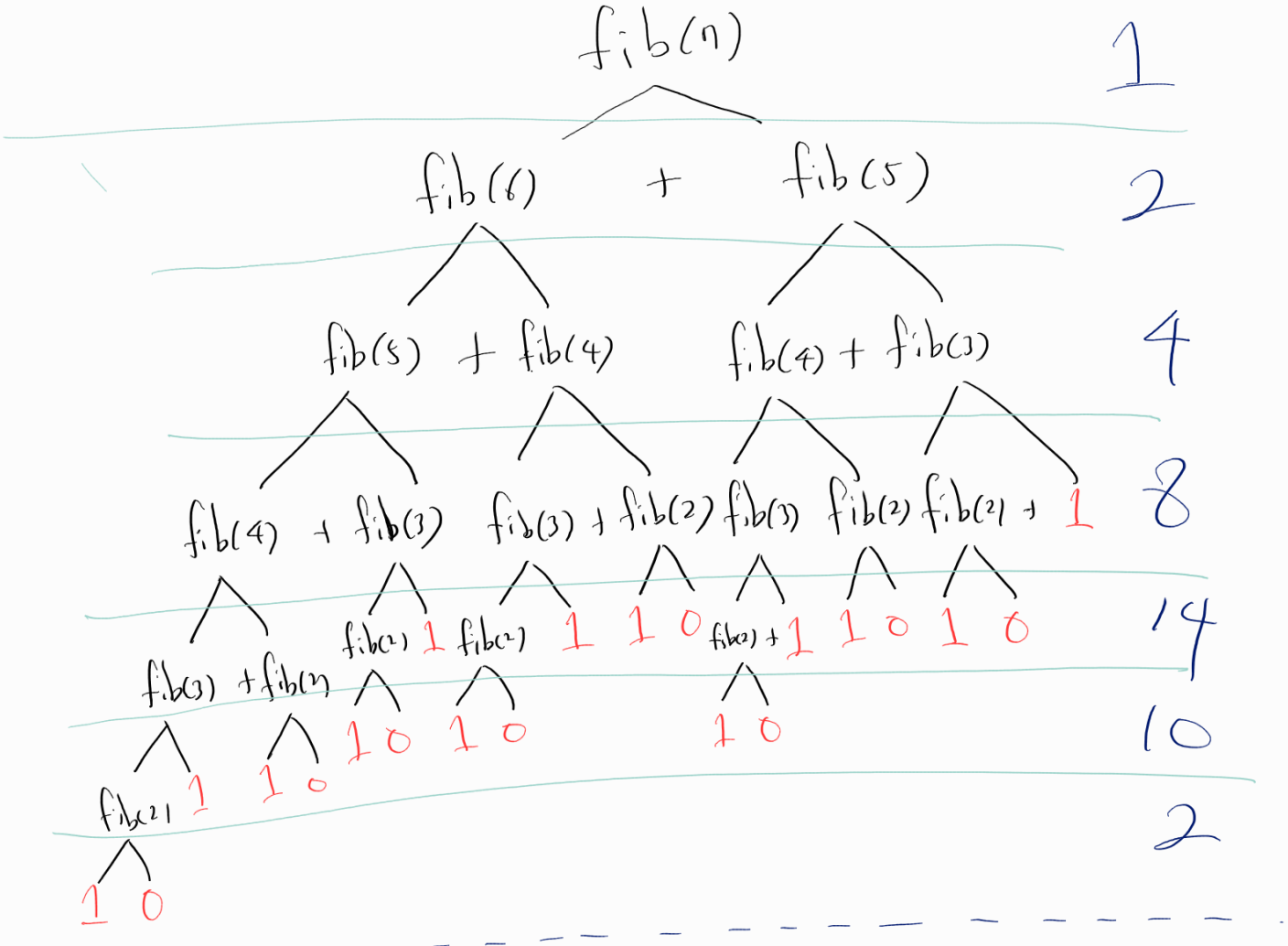
⇒ temp=13, last=8, current=13

∴ fib(7)=13 //

POA 276. function fib(n)

comment Return Fn, the nth Fibonacci number.

1. if n≤1 then return(n) # n≤1이면 n은 return
2. else return(fib(n-1) + fib(n-2)) # 아니면 재귀함수 n-1, n-2를 각각 넣어 재귀호출



총 호출 횟수 : 41번

∴ fib(n) = 1×13 + 0×8

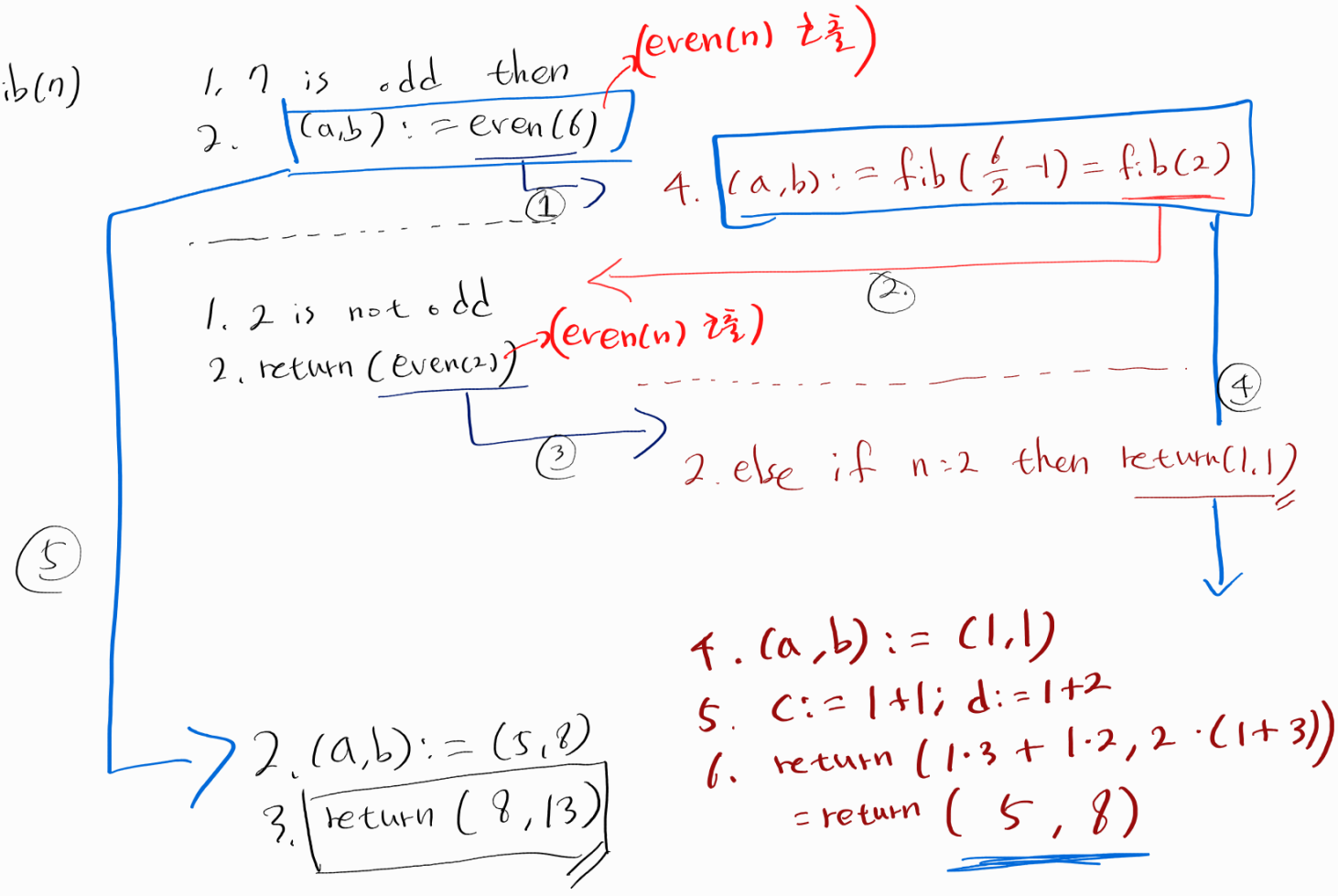
= 13

POA.277

```
function fib(n)
  comment Return (Fn-1, Fn)
1. if n is odd then
2.     (a,b) := even(n-1)
3.     return (b, a+b)
4. else return (even(n))
```

```
function even(n)
  comment Return (Fn-1, Fn) when n is even
1. if n=0 then Return(1,0)
2. else if n=2 then Return(1,1)
3. else if n=4 then Return(2,3)
4. (a,b) := fib(n/2 - 1)
5.    c := a+b; d := b+c
6. return (b·d + a·c, c·(d+b))
```

∴ fib(7)

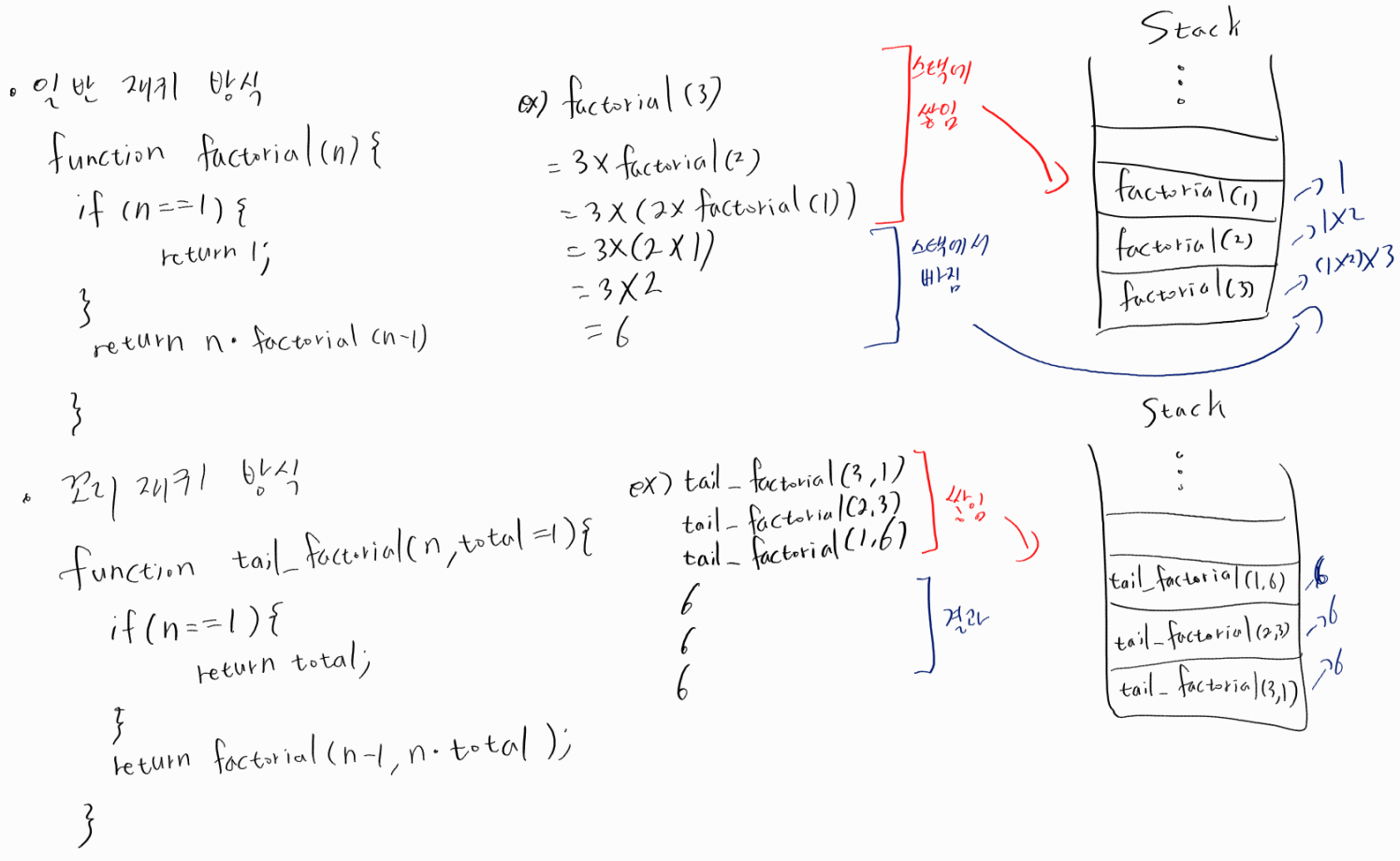


∴ fib(7) = (8, 13) //

even 함수 : 2번 호출

꼬리 재귀(Tail Recursion)

꼬리 재귀에 대해 알아보기 전에, '재귀'의 정확한 정의에 대해 한 번 짚고 넘어가보겠다. 재귀함수란, 한 마디로 '자기 자신을 호출하는 함수'이다. 구조는 비슷하지만 더 작은 문제로 쪼갤 수 있는 문제를 풀 때 유용한 접근 방법이다. 재귀를 사용하면 코드가 짧아지고 내용도 직관적으로 파악할 수 있어 가독성이 높아진다는 장점이 있다. 한 번 호출 될 때마다 함수의 매개변수, 결과값, 그리고 리턴 후 돌아갈 위치 등이 스택에 쌓이게 된다. 허나 재귀는 앞서 말했듯, 호출의 호출의 호출의 호출의...를 반복하는 구조이다보니 '스택 오버 플로우' 현상을 일으킬 수 있다는 위험성을 가지고 있다.
재귀는 위와 같은 이유로 장점이 분명하지만, 자칫하면 위험하기 때문에 주의해서 사용해주어야 하는데, 재귀함수의 장점은 살리고 단점을 보완하는 방법 중 하나가 바로 '꼬리재귀'이다.
꼬리 재귀는 '재귀 호출이 끝나면 아무 일도 하지 않고 결과만 바로 반환되도록 하는 방법'이다. 이 방식을 사용하면 이전 함수의 상태를 유지하지도 않고 추가 연산을 하지도 않아서 스택이 넘쳐나는 문제를 해결할 수 있게 된다. 꼬리 재귀 함수는 이름처럼 항상 함수의 꼬리부분에서 실행되는데, return 되기 전에 값이 정해지며 호출당한 함수의 결과값이 호출하는 함수의 결과값으로 반환된다.
예를 들어보겠다.



둘은 언뜻 비슷해보일 수 있지만, 가장 큰 차이는 값들을 반환할 때 나타난다. 그림을 보면 가장 마지막으로 계산된 total 값인 6이 그대로 return 되는 것을 알 수 있다. 각 함수들은 별도의 연산 없이 이 값을 "전달 만 하는 것"이다. 일반 재귀 함수는 값을 받으면, "그 값에 연산을 하고 다른 함수에게 전달을 해줘야 했다." 하지만 꼬리재귀는 "아무것도 하지 않고 값을 전달"한다.

알아보다보니 오늘 푼 문제 중, POA 277번과 비슷하다고 생각했는데, 꼬리재귀함수에 대해 찾아보니 POA 277번이 꼬리재귀함수라는 것을 깨달았다. 갑자기 이유없이 꼬리재귀함수를 알아오라고 말씀하실 교수님이 아니실텐데... 싶었는데 역시 오늘 푼 문제와 연계해서 조사하고 생각해보라는 교수님의 깊은(?) 뜻이 아니신가 싶다.
* 출처 : https://joooing.tistory.com/m/entry/%EC%9E%AC%EA%B7%80-%E2%86%92-%EA%BC%AC%EB%A6%AC-%EC%9E%AC%EA%B7%80-Tail-Recursion *