ILE3-019 재귀 함수

자기 자신을 다시 호출하여 문제를 해결하는 방법

이전의 값을 사용하는 경우 간단하게 구현이 가능 하며, 분할정복 알고리즘에 적합

피보나치 수열

0과 1로 시작하고 n번째 피보나치 수는 바로 직전의 두 피보나치 수의 합

```
0,  1,  1,  2,  3,  5,  8,  13,  21,  34,  55,  89,  144, 233, ......
```

피보나치 수열은 아래와 3번째 이 후의 값은 아래 와 같이 계산된다

```
0 + 1 = 1
1 + 1 = 2
1 + 2 = 3
2 + 3 = 5
```

위의 값을 수식으로 표현하면 n번째 값은 아래와 같다

```
f(n) = f(n-1) + f(n-2)
```

Python으로의 구현

```
def fibonacci(n):
    if n <= 0:
        return None
    elif n == 1:
        return 0
    elif n == 2:
        return 1

    prev_fib = 0
    current_fib = 1
    for _ in range(3, n + 1):
        next_fib = prev_fib + current_fib
        prev_fib, current_fib = current_fib, next_fib

    return current_fib
```

재귀함수로 구현

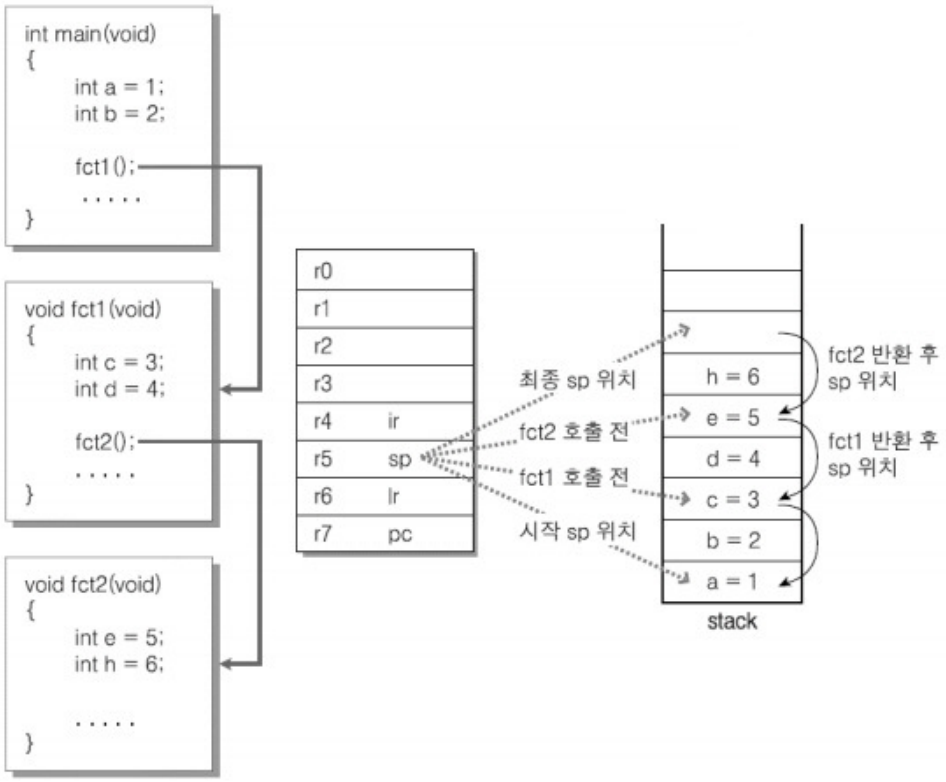
```
def fibonacci(n):
    if n <= 0:
        return None
    elif n == 1:
        return 0
    elif n == 2:
        return 1

    return fibonacci(n - 1) + fibonacci(n - 2)
```

사용 시 주의 사항

함수를 중첩하여 호출하는 구조로 인하여 Process의 Stack의 고갈이 일어 날 수 있어서 너무 깊은 단계의 호출 시 오류가 발생 할 수 있음

Stack

함수 호출 시 전달할 매개변수 및 함수 호출 후 복귀할 위치 및 리턴 값을 임시로 저장한 공간
Process 시작 시 고정된 용량이 할당 되며 일반적으로 16KB ~ 32KB의 용량을 지님



과제

분할정복 알고리즘이었던 합병 정렬을 구현하고 0~1.000,000 범위 내에서 100,000개의 값을 추출하여 정렬 하세요