Exercise 25121-0823. 재귀함수 연습 단계3

Section 1 - 재귀함수 종합 연습 (중급포함)

1. 다음 재귀함수를 제어문만 사용하여 하나의 함수로 구현하세요.

```
def cherry_recursive(n, current=1):
    if current > n:
        return
    print('*' * current)
    cherry_recursive(n, current + 1)

def cherry(n):
    # 여기에 코드를 작성하세요
    pass
```

입력예시

```
5
```

출력예시

```
*
**
***
****
*****
```

해설

```
def cherry(n):
    for i in range(1, n + 1):
        print('*' * i)
```

2. 다음 재귀함수를 제어문만 사용하여 하나의 함수로 구현하세요.

```
def grape_recursive(a, b):
    if b == 0:
        return a
    return grape_recursive(b, a % b)

def grape(a, b):
    # 여기에 코드를 작성하세요
    pass
```

입력예시

```
a=48, b=18
```

출력예시

```
6
```

해설

```
def grape(a, b):
    while b != 0:
        temp = b
        b = a % b
        a = temp
    return a
```

3. 다음 재귀함수를 제어문만 사용하여 하나의 함수로 구현하세요.

```
def elderberry_recursive(n):
    if n <= 0:
        return
    elderberry_recursive(n - 1)
    print(str(n) * n)

def elderberry(n):
    # 여기에 코드를 작성하세요
    pass
```

입력예시

```
5
```

출력예시

```
1
22
333
4444
55555
```

해설

```
def elderberry(n):
    for i in range(1, n + 1):
        print(str(i) * i)
```

4. 재귀함수를 사용하여 조합 nCr을 계산하는 함수를 작성하세요. 조합의 점화식: nCr = (n-1)C(r-1) + (n-1)Cr 기저 조건: nC0 = 1, nCn = 1

```
def combination(n, r):
    # 여기에 코드를 작성하세요
    pass

# 테스트
```

입력예시

```
n=5, r=2
```

출력예시

```
10
```

해설

```
def combination(n, r):
    # 기저 조건
    if r == 0 or r == n:
        return 1

    # 재귀 호출: nCr = (n-1)C(r-1) + (n-1)Cr
    return combination(n - 1, r - 1) + combination(n - 1, r)
```

5. 재귀함수를 사용하여 피보나치 수열의 n번째 값을 계산하는 함수를 작성하세요. 피보나치 수열: F(0)=0, F(1)=1, F(n)=F(n-1)+F(n-2) (n≥2) 예: 0, 1, 1, 2, 3, 5, 8, 13, 21, ...

```
def fibonacci(n):
    # 여기에 코드를 작성하세요
    pass

# 테스트
```

입력예시

```
6
```

출력예시

```
8
```

해설

```
def fibonacci(n):
    # 기저 조건: F(0) = 0, F(1) = 1
    if n == 0:
        return 0
    if n == 1:
        return 1

    # 재귀 호출: F(n) = F(n-1) + F(n-2)
    return fibonacci(n - 1) + fibonacci(n - 2)

# 동작 과정 예시 (fibonacci(5)):
#                   fib(5)
#                  /      \
#               fib(4)    fib(3)
#               /    \    /     \
#           fib(3) fib(2) fib(2) fib(1)
#           /  \   /  \   /  \     |
#       fib(2) fib(1) fib(1) fib(0) fib(1) fib(0)  1
#       /  \     |     |     |     |     |
#   fib(1) fib(0) 1     1     0     1     0
#     |     |
#     1     0
#
# 결과: fib(5) = 5

# 참고: 이 방법은 같은 값을 여러 번 계산하므로 비효율적입니다.
# 실제로는 메모이제이션이나 동적 프로그래밍을 사용해야 합니다.
```

6. 다음 제어문으로 구현된 함수를 재귀함수로 변환하세요.

```
# 제어문 구현
def apple_iterative(char, n):
    for i in range(n):
        print(char, end='')

# 재귀함수로 변환하여 구현하세요
def apple(char, n):
    # 여기에 코드를 작성하세요
    pass
```

입력예시

```
char='*', n=5
```

출력예시

```
*****
```

해설

```
def apple(char, n):
    if n <= 0:
        return
    print(char, end='')
    apple(char, n - 1)
```

7. 다음 재귀함수를 제어문만 사용하여 하나의 함수로 구현하세요.

```
def apple_recursive(char, n):
    if n <= 0:
        return
    print(char, end='')
    apple_recursive(char, n - 1)

def apple(char, n):
    # 여기에 코드를 작성하세요
    pass
```

입력예시

```
char='*', n=5
```

출력예시

```
*****
```

해설

```
def apple(char, n):
    for i in range(n):
        print(char, end='')
```

8. 재귀함수를 사용하여 주어진 양의 정수를 뒤집은 값을 반환하는 함수를 작성하세요. 예: 1234 → 4321, 567 → 765

```
def reverse_number(n, reversed_n=0):
    # 여기에 코드를 작성하세요
    pass
```

입력예시

```
1234
```

출력예시

```
4321
```

해설

```
def reverse_number(n, reversed_n=0):
    # 기저 조건: n이 0이면 뒤집은 결과 반환
    if n == 0:
        return reversed_n

    # 재귀 호출: 마지막 자릿수를 reversed_n에 추가하고,
    # n에서 마지막 자릿수를 제거
    return reverse_number(n // 10, reversed_n * 10 + n % 10)

# 더 이해하기 쉬운 버전 (내부 함수 사용):
def reverse_number_v2(n):
    def helper(num, result):
        if num == 0:
            return result
        return helper(num // 10, result * 10 + num % 10)

    return helper(n, 0)
```

9. 재귀함수를 사용하여 문자열의 길이를 계산하는 함수를 작성하세요. (len() 함수를 사용하지 않고) 예: "hello"의 길이는 5

```
def string_length(s):
    # 여기에 코드를 작성하세요
    pass
```

입력예시

```
"hello"
```

출력예시

```
5
```

해설

```
def string_length(s):
    # 기저 조건: 빈 문자열이면 0 반환
    if not s:  # s == "" 와 같음
        return 0
    # 재귀 호출: 1 + (첫 글자를 제거한 문자열의 길이)
    return 1 + string_length(s[1:])
```

10. 재귀함수를 사용하여 주어진 양의 정수의 각 자릿수를 모두 더한 값을 계산하는 함수를 작성하세요. 예: 1234의 경우 1+2+3+4 = 10

```
def sum_of_digits(n):
    # 여기에 코드를 작성하세요
    pass
```

입력예시

```
1234
```

출력예시

```
10
```

해설

```
def sum_of_digits(n):
    # 기저 조건: n이 0이면 0 반환
    if n == 0:
        return 0
    # 재귀 호출: 마지막 자릿수 + 나머지 자릿수들의 합
    return (n % 10) + sum_of_digits(n // 10)
```