Exercise 25121-0823. 재귀함수 연습 단계3

Section 1 - 재귀함수 종합 연습 (중급포함)

1. 다음 재귀함수를 제어문만 사용하여 하나의 함수로 구현하세요.

```
def lemon_recursive(n, base):
    if n == 0:
        return "0"
    if n < base:
        if n < 10:
            return str(n)
        else:
            return chr(ord('A') + n - 10)

    remainder = n % base
    if remainder < 10:
        remainder_char = str(remainder)
    else:
        remainder_char = chr(ord('A') + remainder - 10)

    return lemon_recursive(n // base, base) + remainder_char

def lemon(n, base):
    # 여기에 코드를 작성하세요
    pass
```

입력예시

```
n=255, base=16
```

출력예시

```
FF
```

해설

```
def lemon(n, base):
    if n == 0:
        return "0"

    result = ""
    while n > 0:
        remainder = n % base
        if remainder < 10:
            result = str(remainder) + result
        else:
            result = chr(ord('A') + remainder - 10) + result
        n = n // base
    return result
```

2. 재귀함수를 사용하여 별(*)로 역삼각형을 출력하는 함수를 작성하세요. n줄짜리 역삼각형에서 첫 번째 줄에 n개, 마지막 줄에 1개의 별을 출력합니다.

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

입력예시

```
4
```

출력예시

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

해설

```
def print_reverse_stars(n):
    # 기저 조건: n이 0 이하이면 종료
    if n <= 0:
        return
    # 현재 줄에 n개의 별 출력
    print('*' * n)
    # 재귀 호출: n-1개 별의 역삼각형 출력
    print_reverse_stars(n - 1)
```

3. 재귀함수를 사용하여 n부터 1까지 카운트다운하며 출력하는 함수를 작성하세요. 예: n=5일 때 5, 4, 3, 2, 1을 각각 한 줄씩 출력

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

입력예시

```
5
```

출력예시

```
5
4
3
2
1
```

해설

```
def countdown(n):
    # 기저 조건: n이 0 이하이면 종료
    if n <= 0:
        return
    # 현재 숫자를 출력
    print(n)
    # 재귀 호출: n-1부터 카운트다운
    countdown(n - 1)
```

4. 재귀함수를 사용하여 1부터 n까지의 수 중에서 짝수만 더한 값을 계산하는 함수를 작성하세요. 예: n=6일 때 2+4+6 = 12

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

입력예시

```
6
```

출력예시

```
12
```

해설

```
def sum_even(n):
    # 기저 조건: n이 0 이하이면 0 반환
    if n <= 0:
        return 0

    # n이 짝수인지 확인
    if n % 2 == 0:
        # 짝수이면 n을 더하고 재귀 호출
        return n + sum_even(n - 1)
    else:
        # 홀수이면 더하지 않고 재귀 호출
        return sum_even(n - 1)
```

5. 재귀함수를 사용하여 리스트에서 특정 값이 처음 나타나는 인덱스를 찾는 함수를 작성하세요. 값이 없으면 -1을 반환합니다. (선형 탐색의 재귀 구현)

```
def find_index(arr, target, index=0):
    # 여기에 코드를 작성하세요
    pass
```

입력예시

```
arr=[1,3,5,7,3], target=3
```

출력예시

```
1
```

해설

```
def find_index(arr, target, index=0):
    # 기저 조건: 인덱스가 배열 범위를 벗어나면 찾지 못함
    if index >= len(arr):
        return -1

    # 현재 위치의 값이 target과 같은지 확인
    if arr[index] == target:
        return index

    # 다음 인덱스를 재귀적으로 탐색
    return find_index(arr, target, index + 1)

# 슬라이싱을 사용한 버전:
def find_index_slice(arr, target):
    # 빈 배열이면 찾지 못함
    if not arr:
        return -1

    # 첫 번째 원소가 target과 같으면 0 반환
    if arr[0] == target:
        return 0

    # 나머지 부분에서 재귀 탐색
    rest_index = find_index_slice(arr[1:], target)

    # 나머지 부분에서 찾지 못했으면 -1
    if rest_index == -1:
        return -1

    # 나머지 부분에서 찾았으면 인덱스에 1을 더함
    return rest_index + 1

# 마지막 등장 인덱스를 찾고 싶다면:
def find_last_index(arr, target, index=None):
    if index is None:
        index = len(arr) - 1

    if index < 0:
        return -1

    if arr[index] == target:
        return index

    return find_last_index(arr, target, index - 1)
```

6. 재귀함수를 사용하여 문자열에서 특정 문자가 몇 개 있는지 세는 함수를 작성하세요. 예: "hello"에서 'l'은 2개

```
def count_char(s, c):
    # 여기에 코드를 작성하세요
    pass
```

입력예시

```
s="hello", c='l'
```

출력예시

```
2
```

해설

```
def count_char(s, c):
    # 기저 조건: 빈 문자열이면 0 반환
    if not s:
        return 0
    # 첫 글자가 찾는 문자와 같은지 확인
    if s[0] == c:
        # 같으면 1을 더하고 나머지 문자열 확인
        return 1 + count_char(s[1:], c)
    else:
        # 다르면 0을 더하고 나머지 문자열 확인
        return 0 + count_char(s[1:], c)
```

7. 다음 제어문으로 구현된 함수를 재귀함수로 변환하세요.

```
def lemon_iterative(base, exponent):
    result = 1
    for i in range(exponent):
        result *= base
    return result

def lemon(base, exponent):
    # 여기에 코드를 작성하세요
    pass
```

입력예시

```
base=2, exponent=5
```

출력예시

```
32
```

해설

```
def lemon(base, exponent):
    if exponent == 0:
        return 1
    return base * lemon(base, exponent - 1)
```

8. 재귀함수를 사용하여 1부터 n까지의 수 중에서 홀수만 더한 값을 계산하는 함수를 작성하세요. 예: n=6일 때 1+3+5 = 9

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

입력예시

```
6
```

출력예시

```
9
```

해설

```
def sum_odd(n):
    # 기저 조건: n이 0 이하이면 0 반환
    if n <= 0:
        return 0

    # n이 홀수인지 확인
    if n % 2 == 1:
        # 홀수이면 n을 더하고 재귀 호출
        return n + sum_odd(n - 1)
    else:
        # 짝수이면 더하지 않고 재귀 호출
        return sum_odd(n - 1)
```

9. 다음 재귀함수를 제어문만 사용하여 하나의 함수로 구현하세요.

```
def kiwi_recursive(n):
    if n == 0:
        return "0"
    if n == 1:
        return "1"
    return kiwi_recursive(n // 2) + str(n % 2)

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

입력예시

```
10
```

출력예시

```
1010
```

해설

```
def kiwi(n):
    if n == 0:
        return "0"

    result = ""
    while n > 0:
        result = str(n % 2) + result
        n = n // 2
    return result
```

10. 다음 제어문으로 구현된 함수를 재귀함수로 변환하세요.

```
def fig_iterative(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

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

입력예시

```
5
```

출력예시

```
120
```

해설

```
def fig(n):
    if n <= 1:
        return 1
    return n * fig(n - 1)
```