

```
'''
```

A particular recursive function
and a particular call of this function are given.
Determine, using only pen and paper,
what will be the total number of calls of this function
after the given function call is issued.
Explain your reasoning.

```
'''
```

```
def func( n ):  
    if n <= 0: return 1  
    if n <= 2: return 2  
    value = func( n-1 ) + func( n-2 )  
    return value
```

```
#the call:  
print( func(6) )
```

```
'''
```

A particular recursive function
and a particular call of this function are given.
Determine, using only pen and paper,
what will be the total number of calls of this function
after the given function call is issued.
Explain your reasoning.

```
'''
```

```
def funcx( n, Max ):  
    if n > Max: return 1  
    if n < 0: return 0  
    value = funcx( n+1, Max ) + funcx( n+2, Max )  
    return value
```

```
#the call:  
print( funcx(2,7) )
```

```
'''  
Determine, using only pen and paper,  
the return value and the number of calls  
of the recursive function recur  
when the given code is executed.  
Explain your reasoning.  
'''
```

```
def recur( arr, i ):  
    if i > len( arr ): return 0  
    if arr[i] == 0: return 0  
    x = arr[i] + recur( arr, i + 1 )  
    return x
```

```
arr = [2,3,4,5,0,0,1,2]  
result = recur( arr, 0 )  
print(result)
```

```
'''
```

A particular recursive function
and a particular call of this function are given.
Determine, using only pen and paper,
what will be the total number of calls of this function
after the given function call is issued.
Explain your reasoning.

```
'''
```

```
def funcxx( n, Low ):  
    if n < Low: return 1  
    if n < 10: funcxx( n-1, Low )  
    value = funcxx( n-1, Low ) + funcxx( n-3, Low )  
    return value
```

```
#the call:  
print( funcxx(15, 5) )
```

```
'''
```

A particular recursive function
and a particular call of this function are given.
Determine, using only pen and paper,
what will be the total number of calls of this function
after the given function call is issued.
Explain your reasoning.

```
'''
```

```
def func( n ):  
    if n <= 0: return 1  
    if n <= 1: return func(n-1) + func(n-1)  
    value = func(n-1) + func(n-1) + func(n-1)  
    return value
```

```
#the call:  
print( func (4) )
```

```
'''
```

A particular recursive function
and a particular call of this function are given.
Determine, using only pen and paper,
what will be the total number of calls of this function
after the given function call is issued.
Explain your reasoning.

```
'''
```

```
def func( n ):  
    if n <= 0: return 1  
    if n <= 3: return 2 + func(n-1)  
    value = func(n-2) + func(n-2)  
    return value
```

```
#the call:  
print( func (6) )
```

```
'''
```

A particular recursive function
and a particular call of this function are given.
Determine, using only pen and paper,
what will be the total number of calls of this function
after the given function call is issued.
Explain your reasoning.

```
'''
```

```
def func( array ):  
    for i in range( len(array) ):  
        if array[i] < 0:  
            array[i] += 1  
            func( array )  
    return sum(array)  
  
#the call:  
print( func( [10, -2, -3, 14 ] ) )
```

```
'''
```

A particular recursive function
and a particular call of this function are given.
Determine, using only pen and paper,
what will be the total number of calls of this function
after the given function call is issued.
Explain your reasoning.

```
'''
```

```
def f_recur( n ):  
    if n < 0: return 2  
    if n <= 8:  
        return f_recur(n-2) + 2 * f_recur(n-2)  
    return 10 * f_recur( n-2 )
```

```
#the call:  
print( f_recur( 20 ) )
```