#14
```
def T(n) :
    if n == 1 :
        return 1
    return 7*T(n//5)+10*n

print(T(625))
```

#17
```
def hanoi(n, start, via, des) :
    count = 0
    if n == 1 :
        return 1
    else :
        count += hanoi(n-1,start,des,via)
        count += 1
        count += hanoi(n-1,via,start,des)
    return count

n = int(input())
print(hanoi(n,1,2,3))
```

#42
```
import numpy
# n 은 2의 제곱수, empty는 트로미노가 빈 좌표 (L자 모형이라 defalt = 1로 둔다.)
count = 1
def tromino(n, array, empty = 1) :
    global count
    # n이 2일 때 타일링
    if n == 2 :
        if empty != 0 :
            array[0][0] = count
        if empty != 1 :
            array[0][1] = count
        if empty != 2 :
            array[1][0] = count
        if empty != 3 :
            array[1][1] = count
        count = count + 1
    else :
        # 중앙의 empty 부분이 빈 사각형 모양
        tromino(n//2,array[n//4:(n//4)*3, n//4:(n//4)*3], empty)
        if empty != 0 :
            tromino(n//2,array[0:n//2, 0:n//2], 3) # 오른쪽 아래가 빈 사각형 모양
        if empty != 1 :
            tromino(n//2,array[:n//2, n//2:], 2) # 왼쪽 아래가 빈 사각형 모양
        if empty != 2 :
            tromino(n//2,array[n//2: , 0:n//2], 1) # 오른쪽 위가 빈 사각형 모양
        if empty != 3 :
            tromino(n//2,array[n//2:, n//2:], 0) # 왼쪽 위가 빈 사각형 모양

m = 8
array = numpy.array([[0]*m for _ in range(m)])
tromino(m, array,1)
for arr in array :
    for i in arr :
        print("%3d"%i, end = " ")
    print()
```

```python
#45
import sys
#Subarray의 합을 구함
def sum_subarray(arr, start_index, end_index) :
  func_type = 1
  if start_index > end_index : # for의 type을 구한다.(왼쪽이냐 오른쪽이냐)
    func_type = -1

  result = -sys.maxsize
  s = 0
  for i in range(start_index, end_index,func_type):
    s = s+arr[i]
    if (s>result):
      result = s
  return result

#중간에서 왼쪽 오른쪽의 Subarray의 최대 합을 구하고 더해서 return
def max_mid_subarray(arr, low, high):
  mid = (high+low)//2

  left_sum = sum_subarray(arr, mid, low-1)
  right_sum = sum_subarray(arr, mid+1, high+1)

  return left_sum+right_sum

#중간을 기준으로 왼쪽 절반, 오른쪽 절반을 재귀 호출, 중간의 최대값을 구함
#이 셋의 max값을 return
def max_subarray(arr, low, high):
  if (high == low):
    return arr[high]

  mid = (high+low)//2

  max_left_subarray = max_subarray(arr, low, mid)
  max_right_subarray = max_subarray(arr, mid+1, high)
  max_middle_subarray = max_mid_subarray(arr, low, high)

  return max(max_left_subarray, max_right_subarray , max_middle_subarray)
```