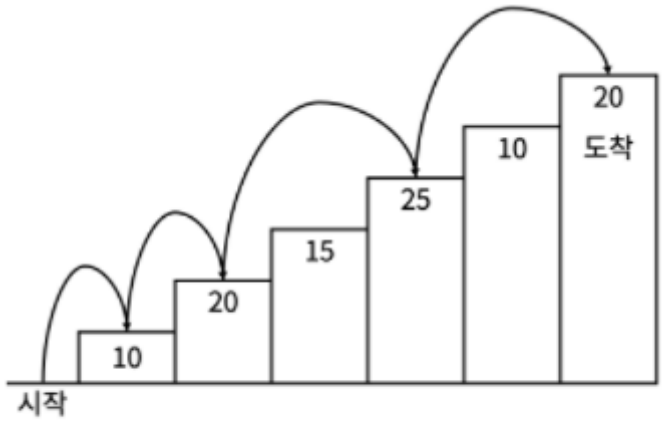
Week 10. DP & 문자열

▼ 2579. 계단 오르기

문제

계단 오르기 게임은 계단 아래 시작점부터 계단 꼭대기에 위치한 도착점까지 가는 게임이다. <그림 1>과 같이 각각의 계단에는 일정한 점수가 쓰여 있는데 계단을 밟으면 그 계단에 쓰여 있는 점수를 얻게 된다.

예를 들어 <그림 2>와 같이 시작점에서부터 첫 번째, 두 번째, 네 번째, 여섯 번째 계단을 밟아 도착점에 도달하면 총 점수는 10 + 20 + 25 + 20 = 75점이 된다.



<그림 2>

계단 오르는 데는 다음과 같은 규칙이 있다.

1. 계단은 한 번에 한 계단씩 또는 두 계단씩 오를 수 있다. 즉, 한 계단을 밟으면서 이어서 다음 계단이나, 다음 다음 계단으로 오를 수 있다.
2. **연속된 세 개의 계단을 모두 밟아서는 안 된다.** 단, 시작점은 계단에 포함되지 않는다.
3. **마지막 도착 계단은 반드시 밟아야 한다.**

따라서 첫 번째 계단을 밟고 이어 두 번째 계단이나, 세 번째 계단으로 오를 수 있다. 하지만, 첫 번째 계단을 밟고 이어 네 번째 계단으로 올라가거나, 첫 번째, 두 번째, 세 번째 계단을 연속해서 모두 밟을 수는 없다.

각 계단에 쓰여 있는 점수가 주어질 때 이 게임에서 얻을 수 있는 총 점수의 최댓값을 구하는 프로그램을 작성하시오.

- **입력**
  - 입력의 첫째 줄에 **계단의 개수**가 주어진다.
  - 둘째 줄부터 한 줄에 하나씩 **제일 아래에 놓인 계단부터 순서대로 각 계단에 쓰여 있는 점수**가 주어진다.
  - 계단의 개수는 300이하의 자연수이고, 계단에 쓰여 있는 점수는 10,000이하의 자연수이다.
- 출력
  - 첫째 줄에 계단 오르기 게임에서 얻을 수 있는 **총 점수의 최댓값**을 출력한다.

입력

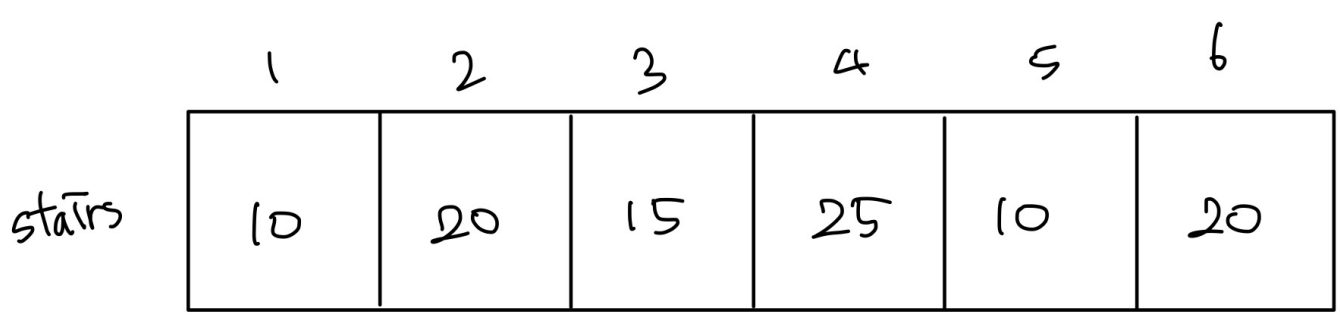
```
6
10
20
15
25
```

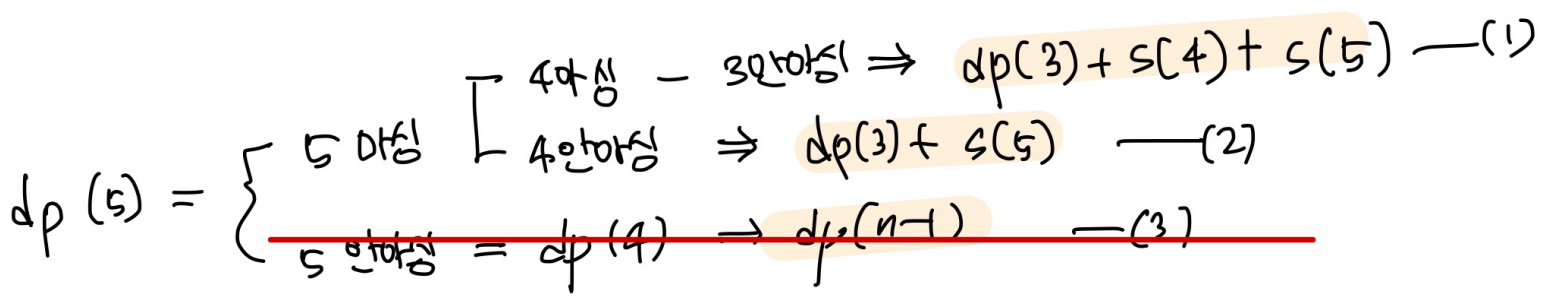
```
10
20
```

출력

```
75
```

구현 아이디어





코드

```
import sys
input = sys.stdin.readline

N = int(input())
stairs = [0]

for _ in range(N):
    stairs.append(int(input()))

memo = [0] * (N+1)
memo[1] = stairs[1]
memo[2] = stairs[1] + stairs[2]
if N == 1:
    print(stairs[1])
elif N == 2:
    print(stairs[2])
else:
    for i in range(3, N+1):
        memo[i] = max(memo[i-2], stairs[i-1]+memo[i-3]) + stairs[i]

    print(memo[-1])
```

▼ 2293. 동전 1

n가지 종류의 동전이 있다. 각각의 동전이 나타내는 가치는 다르다. 이 동전을 적당히 사용해서, 그 가치의 합이 k원이 되도록 하고 싶다. 그 경우의 수를 구하시오. 각각의 동전은 몇 개라도 사용할 수 있다.

사용한 동전의 구성이 같은데, 순서만 다른 것은 같은 경우이다.

- **입력**
  - 첫째 줄에 n, k가 주어진다. (1 ≤ n ≤ 100, 1 ≤ k ≤ 10,000)
  - 다음 n개의 줄에는 각각의 동전의 가치가 주어진다. 동전의 가치는 100,000보다 작거나 같은 자연수이다.
- 출력
  - 첫째 줄에 경우의 수를 출력한다. 경우의 수는 2^31 보다 작다.

입력

```
3 10
1
2
5
```

출력

```
10
```

구현 아이디어

점화식

- $C(i)$: i번째 코인의 가치
- $D(i,k)$: i번째 코인까지 사용하여 k를 만들 수 있는 경우의 수

$$D(i,k) = \begin{cases} D(i-1,k) & \text{if } C(i) > k \\ D(i,k-C(i)) + D(i-1,k) & \text{if } C(i) \le k \end{cases}$$

coin	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
total	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

coin	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2		1	**(1)**=1	**(2)**=2	**(3)**=2	**(4)**=3	**(5)**=3	**(6)**=4	**(7)**=4	**(8)**=5
(total)	**1**	**2**	**2**	**3**	**3**	**4**	**4**	**5**	**5**	**6**

coin	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2		1	**(1)**=1	**(2)**=2	**(3)**=2	**(4)**=3	**(5)**=3	**(6)**=4	**(7)**=4	**(8)**=5
5					1	**(1)**=1	**(2)**=2	**(3)**=2	**(4)**=3	**(5)**=4
(total)	**1**	**2**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**10**

코드

```
N, K = [int(n) for n in input().split()]
coins = [int(input()) for _ in range(N)]

memo = [0] * (K+1)
memo = [0] * (K+1)

memo[0] = 1

for coin in coins:
    for val in range(coin, K+1):
        memo[val] += memo[val-coin]

print(memo[-1])
```

▼ 2133. 타일 채우기

3×N 크기의 벽을 2×1, 1×2 크기의 타일로 채우는 경우의 수를 구해보자.

아래 그림은 3×12 벽을 타일로 채운 예시이다.

- **입력**
  - 첫째 줄에 N(1 ≤ N ≤ 30)이 주어진다.
- 출력
  - 첫째 줄에 경우의 수를 출력한다.

입력

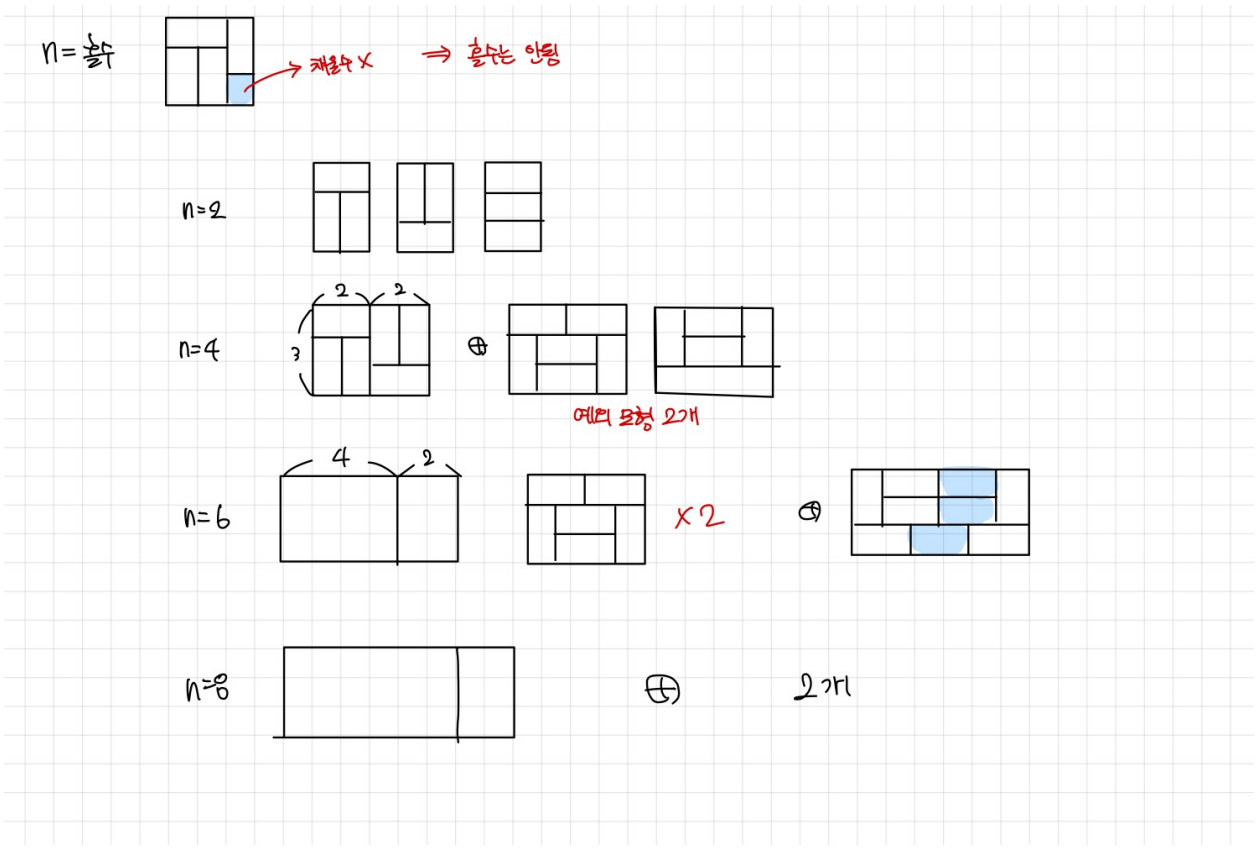
```
2
```

출력

```
3
```

구현 아이디어

```
8
1. mono[4] = 3 * memo[2] = 9
   2. mono[4] += 2 * memo[0] = 2
1. mono[6] = 3 * memo[4] = 33
   2. mono[6] += 2 * memo[2] = 6
   2. mono[6] += 2 * memo[0] = 2
1. mono[8] = 3 * memo[6] = 123
   2. mono[8] += 2 * memo[4] = 22
   2. mono[8] += 2 * memo[2] = 6
   2. mono[8] += 2 * memo[0] = 2
153
```



코드

```
# 2133
import sys
sys.setrecursionlimit(10**6)

N = int(input())
memo = [0] * (31)

memo[0] = 1
memo[2] = 3

if N % 2:
    print(0)
else:
    for n in range(4, N+1, 2):
        # 2x3 도형으로 만들 수 있는 경우
        memo[n] = 3 * memo[n - 2]

        # 예외 경우들에 대한 경우(원본, 180 회전)
        for nxt in range(4, n + 1, 2):
            memo[n] += 2 * memo[n - nxt]

    print(memo[n])
```

▼ 9095. 1, 2, 3, 더하기

정수 4를 1, 2, 3의 합으로 나타내는 방법은 총 7가지가 있다. 합을 나타낼 때는 수를 1개 이상 사용해야 한다.

- 1+1+1+1
- 1+1+2
- 1+2+1
- 2+1+1
- 2+2
- 1+3
- 3+1

정수 n이 주어졌을 때, n을 1, 2, 3의 합으로 나타내는 방법의 수를 구하는 프로그램을 작성하시오.

- **입력**
  - 첫째 줄에 테스트 케이스의 개수 T가 주어진다.
  - 각 테스트 케이스는 한 줄로 이루어져 있고, 정수 n이 주어진다. n은 양수이며 11보다 작다.
- 출력
  - 각 테스트 케이스마다, n을 1, 2, 3의 합으로 나타내는 방법의 수를 출력한다.

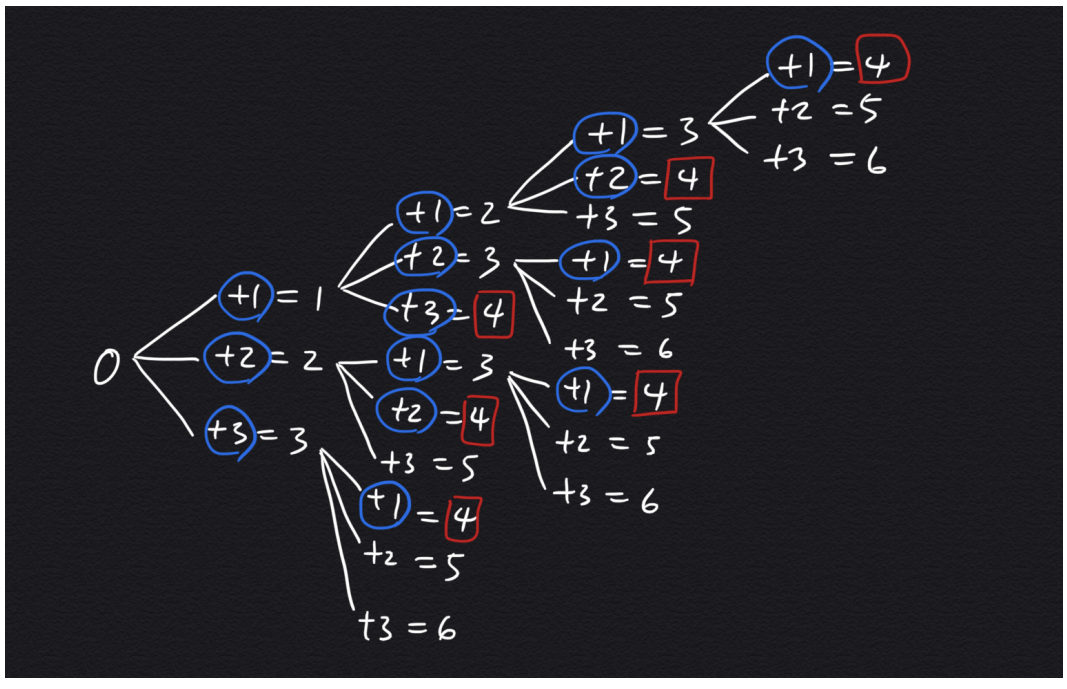
입력

```
3
4
7
10
```

출력

```
7
44
274
```

구현 아이디어



코드

```
def make_case(target, sums):
    if sums == target:
        return 1
    elif sums > target:
        return 0

    result = 0
    for num in nums:
        result += make_case(target, sums + num)

    return result


N = int(input())
cases = [int(input()) for _ in range(N)]

nums = [1, 2, 3]
for case in cases:
    print(make_case(case, 0))
```