#18429 근손실

문제

웨이트 트레이닝을 좋아하는 어떤 대학원생은, 현재 3대 운동 중량 500의 괴력을 소유하고 있다. 다만, 하루가 지날 때마다 중량이 K만큼 감소한다. 예를 들어 K=4일 때, 3일이 지나면 중량이 488로 감소하게 된다. 따라서 운동을 하지 않고, 가만히 있다면 매일매일 중량이 감소할 뿐이다.

다행히도 이 대학원생은 N개의 서로 다른 운동 키트를 가지고 있다. 이 대학원생은 하루에 1개씩의 키트를 사용하며, 매일 어떤 키트를 사용할 지는 마음대로 결정할 수 있다. 운동 키트들은 각각의 중량 증가량을 가지고 있으며, 사용할 때마다 즉시 중량이 증가하게 된다. 이 때 몇몇 운동 키트들의 중량 증가량이 같을 수 있으나, 서로 다른 운동 키트로 간주한다. 각각의 운동 키트는 N일 동안 한 번씩만 사용할 수 있다.

대학원생은 운동 기간동안 항상 중량이 500 이상으로 유지가 되도록 N일간의 운동 플랜을 세우고자 한다. 1일차부터 N일차까지의 모든 기간동안, 어떤 시점에서라도 중량이 500보다 작아지지 않도록 해야 한다.

예를 들어 N=3, K=4일 때, 각 운동 키트의 중량 증가량이 다음과 같다고 가정하자.

운동 키트 번호	1	2	3
중량 증가량	3	7	5

이 때 1번, 3번, 2번 순서대로 운동 키트를 적용한다고 해보자. 이 경우 운동 1일차에 대학원생은 중량이 3만큼 증가하지만 그와 동시에 하루에 중량이 4만큼 감소하기 때문에, 1일이 지난 이후에 중량은 499가 된다. 따라서 조건을 만족하지 못한다.

반면에 3번, 1번, 2번 순서대로 운동 키트를 적용한다고 해보자. 그러면 1일차부터 운동을 모두 마친 날까지의 모든 시점에 대하여 항상 중량이 500이상이 된다.

N개의 운동 키트에 대한 정보가 주어졌을 때, N일간 하루에 1개씩의 운동 키트를 사용하는 모든 경우 중에서, 운동 기간동안 항상 중량이 500 이상이 되도록 하는 경우의 수를 출력하는 프로그램을 작성하시오.

위 예시에서는 모든 경우 중에서 총 4가지 경우가 조건을 만족한다.

운동 키트 적용 순서	조건 만족 여부
1번 – 2번 – 3번	X
1번 – 3번 – 2번	X
2번 – 1번 – 3번	O
2번 – 3번 – 1번	O
3번 – 1번 – 2번	O
3번 – 2번 – 1번	O

입력

첫째 줄에 자연수 N과 K가 공백을 기준으로 구분되어 주어진다. ($1 \le N \le 8$, $1 \le K \le 50$) 둘째 줄에 각 운동 키트의 중량 증가량 A가 공백을 기준으로 구분되어 주어진다. ($1 \le A \le 50$)

출력

N일 동안 N개의 운동 키트를 사용하는 모든 경우 중에서, 운동 기간동안 항상 중량이 500 이상이 되도록 하는 경우의 수를 출력한다.

매일 한개의 운동키트로 운동. 운동키트는 중량이 같더라도 서로 다르게 인식된다.

어떤 순서로 운동키트를 사용할 것인지는 자율이다.

운동을 하면 중량이 증가. 중량은 다음날이 되면 K만큼 손실된다.

모든 운동기간동안 (N일 동안) 항상 중량이 500이상이 되도록하는 경우의 수를 구하시오.

분석

운동키트를 어떤 순서로 사용할 지 모두 구해야하고, 각 경우가 항상 중량 500이상을 만드는지 계산해야한다.

이 때, 운동키트는 배치 순서에 따라 상황이 달라지므로, **순열**을 사용할 수 있다.

순열 구현 방법 1

```
vector<char>arr={'a','b','c','d'};

void swap(int*a ,int* b){
    int temp = *a;
    *a=*b;
    *b=temp;
}

void permutation(int now, int n, int r){
    if(now ==r){
        // 경우의 수 하나 완성
        for(int i=0;i<r;i++){
          cout << arr[i] ;
        }
        cout<<'\n';
    }

    for(int i=now;i<n;i++){
        swap(&arr[now],&arr[i]);
        permutation(now+1,n,r);
        swap(&arr[now],&arr[i]);
    }

}

int main(){
  permutation(0,4,3); // 예제 : 4개중 3개 순서있게 나열
  return 0;
}
```

📒 정리

1. 기준으로 잡을 인덱스의 값(now)을, i 번째 인덱스와 swap으로 뒤바꿔줌.
2. 기준 인덱스(now) 뒤의 요소들을 다시 순열로 배치.
3. 그 후 swap을 원래대로 되돌려 놓기.

배열{'a','b','c','d'}이고, ${}_4P_3$ 인경우

(now ==0)

- a↔a → b,c,d 순열 계산 → a↔a
  - ▼ (now ==1)
    - b↔b → c, d 순열 계산 → b↔b
      - ▼ (now==2)
        - c ↔ c → d 순열 계산 → c ↔ c
          - ▼ (now==3)
            - now == r 이므로 원하는 만큼 순열 완성. 재귀 종료.
        - c ↔ d → c 순열 계산 → d ↔ c
    - b↔c → b, d 순열 계산 → c↔b
    - b↔d → c, b 순열 계산 → d↔b
- a↔b → a,c,d 순열 계산 → b↔a
- a↔c → b,a,d 순열 계산 → c↔a
- a↔d → b,c,a 순열 계산 → d↔a

순열 구현 방법 2

```
vector<int> arr={'a','b','c','d','e'};
vector<int> marking;
vector <int> perm;

void permutation(int now, int n, int r){
```

```
    if(now ==r){
         // 경우의 수 하나 완성
        for(int i=0;i<r;i++){
          cout << perm[i] ;
        }
        cout<<'\n';
    }

    int i;
    for(i=0;i<n;i++){

        if(marking[i]==1)
            continue;

        marking[i] = 1;
        perm[now]= arr[i];
        permutation(now+1,n,r);
        marking[i] = 0;
    }

}
```

문제 풀이 코드 1

```
#include <iostream>
#include <vector>
using namespace std;

int N,K;
int i,j;
int cnt=0;
int weight=500;
vector<int> kit;

void permutation(int now, int n, int r);
void swap(int*a ,int* b);

int main()
{
    cin >> N >>K;

    for(i=0;i<N;i++){
        int n;
        cin>>n;
        kit.push_back(n);
    }

    permutation(0,N,N);
    cout<<cnt;
```

```
    return 0;
}

void swap(int*a ,int* b){
    int temp = *a;
    *a=*b;
    *b=temp;
}

void permutation(int now, int n, int r){

    if(now ==r){
        int i;
        weight=500;
        for(i=0;i<n;i++){
            if(i!=n-1){
                weight+=kit[i];
                weight-=K;
            }
            if(weight<500)
                break;
        }
        if(weight>=500)
            cnt++;
    }

    int i;
    for(i=now;i<n;i++){
        swap(&kit[now],&kit[i]);
        permutation(now+1,n,r);
        swap(&kit[now],&kit[i]);
    }

}
```

문제풀이 코드 2

```
#include <iostream>
#include <vector>
using namespace std;

int N,K;
int i,j;
int cnt=0;
int weight=500;
vector<int> kit;
vector<int> marking;
vector <int> arr;


void initMarking();
void permutation(int now, int n, int r);
```

```
int main()
{
    cin >> N >>K;

    for(i=0;i<N;i++){
        int n;
        cin>>n;
        kit.push_back(n);
        marking.push_back(0);
        arr.push_back(0);
    }

    permutation(0,N,N);
    cout<<cnt;

    return 0;
}

void initMarking(){
    marking.clear();
    for(i=0;i<N;i++){
        marking.push_back(0);
    }
}

void permutation(int now, int n, int r){

    if(now ==r){
        int i;
        weight=500;
        for(i=0;i<n;i++){
            if(i!=n-1){
                weight+=arr[i];
                weight-=K;
            }
            if(weight<500)
                break;
        }
        if(weight>=500)
            cnt++;
    }

    int i;
    for(i=0;i<n;i++){

        if(marking[i]==1)
            continue;

        marking[i] = 1;
        arr[now]= kit[i];
        permutation(now+1,n,r);
        marking[i] = 0;
    }

}
```