초심자(문자열) – 이니셜

이니셜은 "United States of America" → "USA" 와 같이 각 단어의 머리글자만을 따서 만든 축약어의 일종이다.

사람이름의 경우에도 "Hong-Gill-Dong" → "HGD" 와 같이 이니셜을 만들기도 한다.

쿠딩이는 만나는 사람마다 연락처를 저장하는 습관이 있는데, 연락처에 저장된 사람들의 이름 모두 이니셜로 바꿀려고 한다. 하지만 연락처에 저장된 이름이 규칙성이 없고 양이 너무 많기 때문에 이를 일괄로 이니셜로 바꾸어주는 프로그램을 개발하려고 한다.

입력 :

첫 번째 줄에는 입력 받을 이름의 개수 N을 입력한다.

두 번째 줄부터 문장의 개수만큼 이름을 입력한다.

입력 받을 이름의 조건은 다음과 같다.

- 최대 100글자.
- 영어 알파벳 대문자, 소문자, 공백, 쉼표 (',', 아스키코드 44), 하이픈 ('-', 아스키코드 45)로만 이루어져 있다.
- 각 단어의 머리글자는 항상 대문자이며 그 외의 모든 문자는 모두 소문자이다.
- 각 단어의 머리글자 구분 공백, 쉼표, 하이픈으로만 하며 한 문장에 여러 구분문자들이 포함될 수 있다.

출력 :

입력 받은 영어문장의 이니셜을 순서대로 출력한다.

입력 예시 1	입력 예시 2	입력 예시 3
1 Kim Koo Ding	3 Elon,Reeve-Musk Jeff, Bezos Mark-Elliot, Zuckerberg	5 Moon Jae-In Park Geun, Hye Lee,Myung -Bak Roh Moo Hyun Kim-Dae,Jung
출력 예시 1	출력 예시 2	출력 예시 3
KKD	ERM JB MEZ	MJI PGH LMB RMH KDJ

```
N = int(input()) # 입력받을 이름의 개수 선언 및 입력
S = [] # 입력받을 이름의 리스트 선언

for i in range(N): # 입력받을 이름 N개 입력
    S.append(input().replace(',', ' ').replace('-', ' ').split()) # 구분문자 ','와 '-'을 ' '공백으로 대체 후 공백 문자를 토큰으로 이름을 리스트로 입력받음

'''
입력 예시)
3
Elon,Reeve-Musk
Jeff, Bezos
Mark-Elliot, Zuckerberg

S에 다음과 같이 저장됨)
S = [['Elon', 'Reeve', 'Musk'], ['Jeff', 'Bezos'], ['Mark', 'Elliot', 'Zuckerberg']]

'''

for i in range(N): # 이니셜 출력
    for j in S[i]:
        print(j[0], end = '')
    print()
```

재학생(정렬) – 거인국과 소인국

거인국에 사는 쿠딩이는 소인들이 사는 소인국에 놀러 갔다.

쿠딩이는 소인 몇 명을 집어 입에 물고 있던 아이스크림 막대 위에 소인들을 올려 놓았다.

소인들은 겁에 질려 막대 아래로 뛰어내려 탈출하고 있다. 쿠딩이는 소인들에게 번호를 매겨 가장 마지막에 뛰어내린 소인을 데려갈 생각이다.

길이가 M(1 ≤ M ≤ 1,000,000,000)인 막대 위에 N(1 ≤ N ≤ 100,000)명의 소인들이 각자 다른 위치에 서있다. 소인들은 크기가 매우 작기 때문에 이 문제에서는 소인을 크기가 없는 점으로 생각한다. 각각의 소인의 위치는 X(0 < X < L)좌표로 표시된다.

각각의 소인은 왼쪽, 혹은 오른쪽으로 움직이고 있다. 모두 똑같은 속도로 1초에 한 칸씩 움직인다. 소인들은 움직이는 도중에 서로 부딪히는 경우가 생길 수도 있는데, 이 경우 부딪힌 두 명의 소인은 모두 즉시 이동방향을 바꾸어 다시 움직인다.

소인들이 이동하다가 0 또는 M인 위치에 도달하게 되면, 그 소인은 막대 아래로 뛰어내려 탈출하게 된다. 소인들의 초기상태가 주어졌을 때, 가장 마지막에 뛰어내린 소인의 번호와 그 소인이 뛰어내린 시각을 알아내는 프로그램을 작성하시오.

입력 :

첫째 줄에는 두 정수 M, N이 주어진다.

다음 N개의 줄에는 각 소인의 초기 위치가 주어진다.

초기 위치가 양수로 주어지는 경우 그 소인의 진행방향은 오른쪽이며, 그 값이 소인의 위치가 된다.

초기 위치가 음수로 주어지는 경우 그 소인의 진행방향은 왼쪽이며, 그 값이 소인의 위치가 된다.

예시)

5일 경우 소인은 5인 위치에서 오른쪽으로 움직이고 있다.

-3일 경우 소인은 3인 위치에서 왼쪽으로 움직이고 있다.

소인의 번호는 입력에서 주어지는 순서대로 1, 2, 3, ..., N이다.

출력 :

첫째 줄에 두 정수 S, T를 출력한다.

S는 가장 마지막에 뛰어내리는 소인의 번호이며 T는 그 소인이 뛰어내린 시각이다. 가장 마지막에 뛰어내린 소인이 여러 명인 경우는 없다고 가정한다.

입력 예시 1	입력 예시 2	입력 예시 3
6 1 2	6 1 -2	5 2 4 -3
출력 예시 1	출력 예시 2	출력 예시 3
1 4	1 2	2 3
입력 예시 4	입력 예시 5	입력 예시 6
10 2 8 -9	10 3 1 -3 2	10 4 1 -3 2 -4
출력 예시 4	출력 예시 5	출력 예시 6
1 9	3 9	2 9

```
M, N = map(int, input().split()) # 두 정수 M, N 입력받음

P = [] # 소인의 위치를 저장할 리스트
P.insert(0, 0) # P의 0번째 인덱스를 0으로 초기화
Res = [] # 결과 값 저장할 리스트

Cnt_left = 0 # 왼쪽으로 뛰어내린 소인의 수
Jump_left = 0 # 가장 마지막으로 왼쪽으로 뛰어내린 소인이 걸린 시간
Jump_right = 0 # 가장 마지막으로 오른쪽으로 뛰어내린 소인이 걸린 시간

# 소인의 위치를 N줄 만큼 입력
for i in range(1, N + 1): # 소인의 번호가 1부터 시작하기 때문에
    P.insert(i,int(input()))

for i in range(1, N + 1):
    if P[i] >= 0: # i번 소인의 진행방향이 오른쪽 일때
        Jump_right = max(M - P[i], Jump_right) # 오른쪽으로 뛰어내리는 소인 중 가장 오래걸린 시간 찾기
    else: # i번 소인의 진행방향이 왼쪽 일때
        Cnt_left += 1
        Jump_left = max(abs(P[i]),Jump_left) # 왼쪽으로 뛰어내리는 소인 중 가장 오래걸린 시간 찾기

    Res.insert(i - 1, [i, abs(P[i])]) # Res의 행은 소인의 번호, 열은 그 소인이 뛰어내리는데 걸린 시간
```

```
Res.sort(key=lambda x:x[1]) # Res의 열을 기준으로 오름차순 정렬


if Jump_left > Jump_right: # 왼쪽으로 뛰어내린 소인이 걸린 시간이 오른쪽보다 클 경우
    print(Res[Cnt_left - 1][0], Jump_left)
else: # 오른쪽으로 뛰어내린 소인이 걸린 시간이 왼쪽보다 클 경우
    print(Res[Cnt_left][0], Jump_right)
```

```
#include <stdio.h>
#include <iostream>
```

```
#include <algorithm>
#include <math.h>
#include <vector>
#include <queue>
#include <set>
#include <stack>
#include <map>

using namespace std;

int M, N;
int P[100001]; // 소인의 위치를 저장할 배열
int Cnt_left = 0, Jump_left, Jump_right; // 왼쪽으로 뛰어내린 소인의 수, 가장 마지막으로 왼쪽으로 뛰어내린 소인이 걸린 시간, 가장 마지막으로 오른쪽으로 뛰어내린 소인이 걸린 시간
pair<int, int> Res[100001]; // 결과 값 저장할 Pair 클래스

// 두 번째 인덱스를 기준으로 오름차순 정렬을 하기 위한 함수
bool compare(pair<int, int> &a, pair<int, int> &b){
    return a.second < b.second;
}

int main(){

    ios::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
```

```
    cin >> M >> N; // 두 정수 M, N 입력받음


        // 소인의 위치를 N줄 만큼 입력
    for(int i = 1; i <= N; i++){ // 소인의 번호가 1부터 시작하기 때문에
        int tmp; cin >> tmp;
        P[i] = tmp;
    }


    for(int i = 1; i <= N; i++){


        if(P[i] >= 0){ // i번 소인의 진행방향이 오른쪽 일때
            Jump_right = max(M - P[i], Jump_right); // 오른쪽으로 뛰어내리는 소인 중 가장 오래걸린 시간 찾기
        }
        else { // i번 소인의 진행방향이 왼쪽 일때
            Cnt_left++;
            Jump_left = max(abs(P[i]), Jump_left); // 왼쪽으로 뛰어내리는 소인 중 가장 오래걸린 시간 찾기
        }


        Res[i - 1] = make_pair(i, abs(P[i])); // Res의 행은 소인의 번호, 열은 그 소인이 뛰어내리는 데 걸린 시간


    }
```

```
    sort(Res, Res + N, compare); // Res의 열을 기준으로 오름차순 정렬


    if(Jump_left > Jump_right){ // 왼쪽으로 뛰어내린 소인이 걸린 시간이 오른쪽보다 클 경우
        cout << Res[Cnt_left - 1].first << " " << Jump_left;
    } else { // 오른쪽으로 뛰어내린 소인이 걸린 시간이 왼쪽보다 클 경우
        cout << Res[Cnt_left].first << " " << Jump_right;
    }


        return 0;
}
```