(10점) 초심자 1 - 라면 끓이기

시간제한 : 2초 메모리 제한 : 256MB

n명의 새내기들이 MT에 와서 라면을 끓이려고 한다! 먼저, 편의점에서 L ml짜리 생수를 k개 샀다. 그리고 라면도 c 봉지를 사서 각 봉지를 d개의 조각으로 나누었다. 그리고 옆방에서 계란 p개도 뺏어왔다!

라면 1인분을 끓이려면 생수 nL ml와 라면 한 조각 그리고 계란 np개가 필요하다! 새내기들은 한 명당 몇 인분의 라면을 먹을 수 있을까! 모든 사람은 똑같은 양의 라면을 먹는다.

입력 :

1이상 1000 이하의 정수 n, L, c, d, p, nL, np가 순서대로 주어진다. 각 정수는 공백으로 구분된다.

출력 :

한 사람이 먹을 수 있는 라면이 몇 인분인지 정수로 출력하라.

입력 예시 1
3 4 5 10 8 100 3 1
출력 예시 1
2

입력 예시 2
5 100 10 1 19 90 4 3
출력 예시 2
3

입력 예시 3
10 1000 1000 25 23 1 50 1
출력 예시 3
0

(20점) 초심자 2 - 배열의 합이 0이 안되는 방법

시간제한 : 1초 메모리 제한 : 256MB

n개의 정수로 이루어진 배열 $a_1, a_2, a_3, ..., a_n$이 있다.

다음의 조건을 만족하며 n개의 정수로 이루어진 배열 $b_1, b_2, b_3, \dots, b_n$ 을 만들어야 한다.

- 배열 b는 배열 a을 재배열한 배열이다. 따라서 두 배열에는 같은 원소가 같은 개수만큼 존재한다. 다시 말해, 두 집합 $\{a_1, a_2, a_3, ..., a_n\}$과 $\{b_1, b_2, b_3, \dots, b_n\}$은 동일한 집합이다.
- 모든 k=1, 2, ..., n에 대해서 배열 b의 처음 k개의 원소의 합은 0이 아니다.
  모든 k=1, 2, ..., n에 대해 $b_1 + b_2 + ... + b_k \neq 0$

만약 이러한 배열 b가 존재하지 않는다면 NO를 출력하라.

입력 :

첫째 줄에 테스트 케이스의 수 t ($1 \leq t \leq 1000$)이 주어진다.

각 테스트 케이스의 첫째 줄에는 배열 a의 길이 n ($1 \leq n \leq 50$)이 주어진다.

각 테스트 케이스의 둘째 줄에는 배열 a의 원소 n개가 주어진다. ($-50 \leq a_i \leq 50$)

출력 :

각 테스트 케이스에서 만약 제시된 조건을 만족하는 배열 b가 존재하지 않는다면 NO를 출력하고, 만약 존재한다면 YES를 출력한 후 그다음 줄에 배열 b의 원소들을 출력하라. 만약 여러 가지의 배열 b가 존재한다면 그중 아무거나 출력해도 된다.

입력 예시 1
4 4 1 -2 3 -4 3 0 0 0 5 1 -1 1 -1 1 6 40 -31 -9 0 13 -40
출력 예시 1
YES 1 -2 3 -4 NO YES 1 1 -1 1 -1 YES -40 13 40 0 -9 -31

(20점) 초심자 3 - 비밀이야♡

시간제한 : 1초, 메모리 제한 : 128 MB

경기도의 한 기숙학원, 연애는 물론이고 이성 간의 접촉까지 금지된 상황에서 두 남녀가 몰래 쪽지로 사랑을 속삭이는 것은 흔히 있는 일이다. 그러던 어느 날, 어떤 커플이 쪽지를 주고받은 것을 들켜 학원에서 강제퇴사를 당하는 사건이 발생했다. 위기를 느낀 다른 커플은 만약 쪽지가 발각되더라도 그 내용이 들키지 않도록 암호를 만들기로 했다. 암호를 생성하는 방법은 다음과 같다.

<암호생성 방법>

[단계 1]
1. 영어의 모음 A, E, I, O, U에 대응되는 숫자를 결정한다.
2. 영어의 대문자와 문장 부호를 사용하여 문장을 만든다.
3. 띄어쓰기를 포함하여 문장의 각 문자에 0부터 순서대로 인덱스를 붙이고, 문장에 포함된 모음 각각의 수와 인덱스를 기록한다.
4. 모음에 대응되는 숫자와 해당 모음의 개수, 그리고 그 모음의 인덱스들을 차례로 묶어서 모음의 사전순서대로 나열한다. (단, 문장에 사용되지 않은 모음은 생략한다.)

[단계 2]
1. 문장의 모든 모음을 물음표(?)로 치환한다.
2. 띄어쓰기와 물음표를 포함한 문장의 모든 문자를 아스키코드로 변환하여 순서대로 나열한다.

[단계 3]
단계 1과 2의 결과물을 차례로 나열한다.

위의 규칙에 따라 암호를 생성하는 프로그램을 작성하시오.

입력 :
첫째 줄에 모음 A, E, I, O, U 각각에 대응되는 숫자를 순서대로 입력받는다.
둘째 줄에 암호로 변환할 문장을 입력받는다.

출력 :
생성된 암호를 출력한다.

입출력 예시는 다음 페이지에 있습니다.

입력 예시 1
1 2 3 4 5 ABC DE
출력 예시 1
110215636667326863

입력 예시 2
2 4 6 8 10 I ONLY HAVE EYES FOR YOU
출력 예시 2
2184310121461083218221012363326378768932726386633263896383327063823289636 3

입력 예시 3
11 13 15 17 19 HOW MUCH DO YOU LOVE ME?
출력 예시 3
1321922174110131719251472638732776367723268633289636332766386633277636363

입력 예시 4
10 12 14 16 18 MORE THAN YESTERDAY, LESS THAN TOMORROW
출력 예시 4
1037172812431114221641323437776382633284726378328963838463826863894432766383833284726378328463776382826387

(20점) 초심자 4

시간제한 : 1초, 메모리 제한 : 256 MB

길이가 n인 배열 a가 주어졌을 때, 배열 a에 다음의 동작을 원하는 만큼 수행할 것이다.

$1 \le i, j \le n$ 인 두 인덱스 i, j를 골라 $a_i + a_j$ 가 홀수이면 둘의 위치를 바꾼다.

단, 위치를 바꾸기 전의 배열 a보다 위치를 바꾼 후의 배열 a가 사전통계학적으로 작아야 한다.

이 동작을 수행해서 얻을 수 있는 사전 통계학적으로 가장 작은 배열은?

사전통계학적으로 작은 배열의 정의는 다음과 같다.
배열 x, y에 대해, $x_j = y_j$ 이고 $x_i < y_i$ 인 인덱스 i, j ($1 \le j < i$) 쌍이 하나라도 존재하면 배열 x는 배열 y 보다 사전통계학적으로 작다.

예를 들어, n이 4인 배열 a = {7, 6, 5, 4} 가 있다.
1. i = 1이고, j = 2일 때, $a_i + a_j = 13$으로 홀수이므로, 7과 6의 자리를 바꾼다.
a = {6, 7, 5, 4} 가 되며, 이는 위치를 바꾸기 전보다 사전통계학적으로 작다.

2. i = 4 이고 j = 2 일 때, $a_i + a_j = 11$로 홀수이므로 7과 4의 위치를 바꾼다.
a = {6, 4, 5, 7} 가 되며, 이는 위치를 바꾸기 전보다 사전 통계학적으로 작다.

3. 이러한 방법으로 a = {6, 4, 5, 7} -> a = {5, 4, 6, 7} -> a = {4, 5, 6, 7} 로 바뀐다. {4, 5, 6, 7}에서 숫자의 위치를 바꿔서 사전통계학적으로 더 작은 배열은 만들 수 없으므로 (사전통계학적으로 가장 작은 배열이므로) 4 5 6 7이 출력된다.

입력 :
첫 번째 줄에는 $1 \le n \le 10^5$ 인 정수 n이 입력된다. (n은 배열 a의 크기)
두 번째 줄에는 n개의 배열 a의 원소가 공백으로 구분되어 입력된다.

출력 :
얻을 수 있는 사전통계학적으로 가장 작은 배열을 출력한다.
사전통계학적으로 작은 배열을 만들 수 없을 때에는 입력 배열을 그대로 출력한다.

입력 예시 1
3 4 1 7
출력 예시 1
1 4 7

입력 예시 2
2 1 1
출력 예시 2
1 1

(20점) 초심자 5 - 인공지능 비서 쿠딩이

시간제한 : 2초, 메모리 제한 : 128 MB

할 일이 너무 많아 바쁜 진구는 인공지능 비서 쿠딩이에게 일을 맡기기로 했다. 쿠딩이가 일을 처리하기 위해서는, 할 일의 수 K와 각 일의 우선순위 정보가 필요하다. 쿠딩이는 버퍼에 할 일을 모두 넣어둔 후, 일처리 로직에 따라 해야 할 일을 처리한다.

쿠딩이의 일처리 로직
1. 버퍼의 제일 앞쪽에 놓여있는 일의 우선순위를 확인한다.
2. 만약 버퍼에 남아있는 일들 중, 제일 앞쪽 일의 우선순위보다 우선순위가 높은 일이 있다면, 그 일을 가장 마지막으로 미뤄둔다. 만약 제일 앞에 놓인 일이 우선순위가 가장 높은 일이라면, 바로 그 일을 처리한다. 이 때, 우선순위는 숫자가 클수록 높다.

쿠딩이가 K개의 일을 받았을 때, i번째 일이 몇 번째로 처리될까? 일의 순서는 0부터 센다.

예를 들어, 쿠딩이는 우선순위가 9, 2, 3, 5인 일 4개를 받았다. 우리는 2번 일이 몇 번째로 처리되는지 알아보고 싶다. 쿠딩이가 일을 처리하는 순서는 다음과 같을 것이다.

버퍼	0	1	2	3
우선순위	9	2	3	5

버퍼에 남은 일 중 0번 일의 우선순위가 가장 높으므로 첫 번째로 처리한다.

버퍼	1	2	3	
우선순위	2	3	5	

1번 일의 우선순위는 2인데, 나머지 일들 중 우선순위가 더 높은 일이 있으므로 가장 뒤로 미룬다.

버퍼	2	3	1	
우선순위	3	5	2	

2번 일의 우선순위는 3인데, 나머지 일들 중 우선순위가 더 높은 일이 있으므로 가장 뒤로 미룬다.

버퍼	3	1	2	
우선순위	5	2	3	

버퍼에 남은 일 중 3번 일의 우선순위가 가장 높으므로, 3번 일을 두 번째로 처리한다.
1번 일은 뒤로 미뤄질 것이고, 결국 2번 일은 세 번째로 처리될 것이다.

입력 :
첫 번째 줄에 테스트케이스의 수 T가 주어진다.
각 테스트케이스별로 첫 번째 줄에 쿠딩이가 처리할 일 K와, 처리 순번을 알아볼 일의 번호 i가 주어진다.
$1 \leq K \leq 100$ 이고, $0 \leq i < K$ 이다.
두 번째 줄에는 K개 일의 우선순위 정보가 주어진다.
우선순위는 공백으로 구별되어있고, 각 우선순위 x는 $1 \leq x \leq 9$이다.

출력 :
각 테스트케이스에서 i번 일의 처리 순번을 출력한다.

입력 예시 1
3 1 0 7 4 2 5 6 7 8 8 0 2 2 5 2 2 2 2 2
출력 예시 1
1 2 7