배달의 민족에서는 주문이 발생하면 음식점 근처에 있는 배민 라이더들에게 알림을 보냅니다. 다음 그림은 반경이 500m일 때의 예입니다.



중앙의 파란색 마커: 음식점  
빨간색 별: 알림을 받을 라이더  
초록색 별: 알림을 받지 못하는 라이더

주문이 발생한 음식점의 위치 restaurant, 배민 라이더들의 위치 riders, 반경 k가 매개변수로 주어집니다. 알림을 받을 라이더는 몇 명인지 return 하도록 메서드를 완성해주세요.

제한 사항

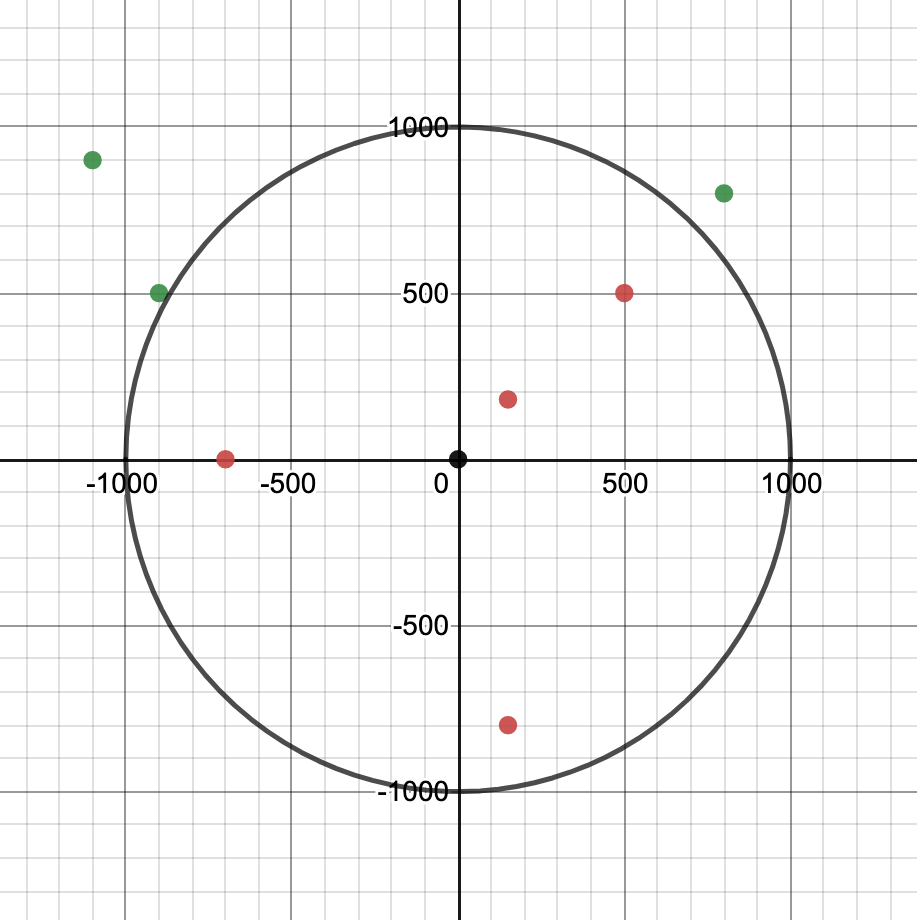
* restaurant 배열의 길이는 2이며, 음식점의 [X좌표, Y좌표] 형식입니다.
* riders 배열의 길이는 1 이상 1,000 이하입니다.
* riders 배열의 원소는 라이더의 [X좌표, Y좌표]형식입니다.
* 모든 X, Y좌표 값은 -10,000 이상 10,000 이하입니다.
* k는 200 이상 5,000 이하입니다.
* 원 경계선에 있는 라이더도 알림을 받습니다.

입출력 예

| **restaurant** | **riders** | **k** | **result** |
| --- | --- | --- | --- |
| [0,0] | [[-700,0], [150,180], [500,500], [150, -800], [800, 800], [-900, 500], [-1100, 900]] | 1000 | 4 |

입출력 예 설명

주어진 좌표를 그림으로 표현하면 다음과 같습니다.



음식점이 있는 원점에서 반경 1000에 위치한 라이더(빨간 점)는 4명입니다.

접속자가 많을 때는 서버를 많이, 접속자가 적을 때에는 서버를 적게 띄워야 합니다. 배달의 민족에서는 각 시간대에 발생한 로그 수를 파악해 미래에 서버를 몇 대 띄울지 계산하려 합니다. 로그는 YYYY/MM/DD hh:mm:ss 형식으로 표시되며, 한국은 UTC+09:00 시간대이므로 로그에 표시된 시각에서 9시를 더해야 올바른 한국 시각이 나옵니다.

예를 들어 2019/10/01 09:33:19 로그는 한국 기준, 2019년 10월 1일 18시 33분 19초에 발생한 로그입니다.

로그 발생 시각을 담은 문자열 logs가 매개변수로 주어졌을 때 0시부터 23시까지, 한국 시간 기준 각 시간대별로 로그가 몇 회씩 발생했는지 return 하도록 메서드를 완성해주세요.

제한 사항

* 발생 시각은 개행(\n) 문자로 구분되어있습니다.
* 발생 시각은 10개 이상 100개 이하로 주어집니다.
* 발생 시각은 YYYY/MM/DD hh:mm:ss 형식이며 24시 표시법을 따릅니다.
  + 연도는 2011 이상 2019 이하 값으로만 주어집니다.
  + 잘못된 시각이 주어지는 경우(-1년 13월 55일 25시 82분 400초 등)는 없습니다.

입출력 예

입력(logs 파라미터 값)

2019/05/01 00:59:19

2019/06/01 01:35:20

2019/08/01 02:01:22

2019/08/01 02:01:23

2019/08/02 03:02:35

2019/10/03 04:05:40

2019/10/04 06:23:10

2019/10/10 08:23:20

2019/10/12 08:42:24

2019/10/23 08:43:26

2019/11/14 08:43:29

2019/11/01 10:19:02

2019/12/01 11:23:10

출력(리턴 값)

[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 1, 1, 0, 1, 0, 4, 0, 1, 1, 0, 0, 0]

입출력 예 설명

각 로그는 다음 시각에 발생했습니다.

| **시각** | **한국시** |
| --- | --- |
| 2019/05/01 00:59:19 | 9시 |
| 2019/06/01 01:35:20 | 10시 |
| 2019/08/01 02:01:22 | 11시 |
| 2019/08/01 02:01:23 | 11시 |
| 2019/08/02 03:02:35 | 12시 |
| 2019/10/03 04:05:40 | 13시 |
| 2019/10/04 06:23:10 | 15시 |
| 2019/10/10 08:23:20 | 17시 |
| 2019/10/12 08:42:24 | 17시 |
| 2019/10/23 08:43:26 | 17시 |
| 2019/11/14 08:43:29 | 17시 |
| 2019/11/01 10:19:02 | 19시 |
| 2019/12/01 11:23:10 | 20시 |

따라서 시간대별 로그가 발생한 횟수는 다음과 같습니다.

| **0시** | **1시** | **2시** | **3시** | **4시** | **5시** | **6시** | **7시** | **8시** | **9시** | **10시** | **11시** | **12시** | **13시** | **14시** | **15시** | **16시** | **17시** | **18시** | **19시** | **20시** | **21시** | **22시** | **23시** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

배달이는 쿠폰을 여러 장 갖고 있습니다. 이 쿠폰을 이용해 최대한 많이 할인을 받으려고 합니다. 한 음식에는 쿠폰을 하나만 적용할 수 있으며, 사용한 쿠폰은 사라집니다.

주문할 음식의 가격 prices, 쿠폰별 할인율 discounts가 매개변수로 주어집니다. 할인을 최대한 많이 받도록 쿠폰을 적용했을 때 배달이는 얼마를 내야 하는지 return 하도록 메서드를 완성해주세요. 이때 소수점은 버립니다.

제한 사항

* prices 배열의 길이는 1 이상 1,000 이하입니다.
* prices 배열의 원소는 5000 이상 150,000 이하인 자연수입니다.
* discounts 배열의 길이는 1 이상 1,000 이하입니다.
* discounts 배열의 원소는 1 이상 100 이하인 자연수입니다.

입출력 예

| **prices** | **discounts** | **result** |
| --- | --- | --- |
| [13000, 88000, 10000] | [30, 20] | 82000 |
| [32000, 18000, 42500] | [50, 20, 65] | 45275 |

입출력 예 설명

입출력 예 #1

13000원짜리 음식에 20% 쿠폰을 적용해 10400원을 결제합니다.  
88000원짜리 음식에 30% 쿠폰을 적용해 61600원을 결제합니다.  
10000원짜리 음식에는 쿠폰을 적용하지 않습니다.

따라서 배달이는 82000원을 내야 합니다.

입출력 예 #2

32,000원짜리 음식에 50% 쿠폰을 적용해 16,000원을 결제합니다.  
18,000원짜리 음식에 20% 쿠폰을 적용해 14,400원을 결제합니다.  
42,500원짜리 음식에 65% 쿠폰을 적용해 14,875원을 결제합니다.

따라서 배달이는 45,275원을 내야 합니다.

배달 앱을 이용하려면 유저는 로그인 → 장바구니에 음식 담기 → 주문 과 같은 순서를 거칩니다. 이를 위해 서버에서는 다음과 같은 일을 합니다.

1. 로그인: LOGIN 아이디 비밀번호
   * 이미 로그인했다면 거부한다.
   * 아이디와 비밀번호가 유효하면 로그인을 허용한다.
   * 아이디와 비밀번호가 유효하지 않으면(아이디가 없거나 비밀번호가 다른 경우) 로그인을 거부한다.
2. 장바구니에 음식 담기: ADD 음식아이디
   * 로그인한 유저라면 허용한다.
   * 로그인하지 않았으면 거부한다.
3. 주문하기: ORDER
   * 장바구니에 담은 음식이 있으면 허용한다. 주문 후에는 장바구니를 비운다.
   * 장바구니에 담은 음식이 없으면 거부한다.

서버에 저장된 유저 정보 infos와 한 사람의 행동을 담은 배열 actions가 매개변수로 주어질 때, 각 행동이 허용되었으면 true, 거부되었으면 false를 담아 return 하도록 solution 메서드를 완성해주세요.

제한 사항

* infos의 길이는 1 이상 10 이하입니다.
* infos의 원소는 아이디 비밀번호 형태입니다.
* actions의 길이는 1 이상 100 이하입니다.
* actions의 원소는 다음 형태 중 하나입니다.
  + LOGIN 아이디 비밀번호
  + ADD 음식아이디
  + ORDER
* infos와 actions 배열에서 주어지는 아이디와 비밀번호는 영문 소문자로만 이루어져 있습니다.
* actions 배열에서 주어지는 음식아이디는 숫자로만 주어집니다.

입출력 예

입력1

infos = ["kim password", "lee abc"]

actions = [

"ADD 30",

"LOGIN kim abc",

"LOGIN lee password",

"LOGIN kim password",

"LOGIN kim password",

"ADD 30",

"ORDER",

"ORDER",

"ADD 40",

"ADD 50"

]

출력1(리턴 값)

[false, false, false, true, false, true, true, false, true, true]

설명1

| **행동** | **결과** | **설명** |
| --- | --- | --- |
| ADD 30 | 거부 | 로그인하지 않았으므로 장바구니에 음식을 담을 수 없습니다. |
| LOGIN kim abc | 거부 | 아이디가 kim인 유저가 있지만, 비밀번호가 다르므로 로그인할 수 없습니다. |
| LOGIN lee password | 거부 | 아이디가 lee인 유저가 있지만, 비밀번호가 다르므로 로그인할 수 없습니다. |
| LOGIN kim password | 허용 | 아이디와 비밀번호가 일치하므로 로그인할 수 있습니다. |
| LOGIN kim password | 거부 | 이미 로그인했으므로 거부합니다. |
| ADD 30 | 허용 | 로그인 한 유저이므로 장바구니에 음식을 담을 수 있습니다. |
| ORDER | 허용 | 장바구니에 담은 음식이 있으므로 주문할 수 있습니다. 장바구니를 비웁니다. |
| ORDER | 거부 | 장바구니에 담은 음식이 없으므로 주문할 수 없습니다. |
| ADD 40 | 허용 | 로그인 한 유저이므로 장바구니에 음식을 담을 수 있습니다. |
| ADD 50 | 허용 | 로그인 한 유저이므로 장바구니에 음식을 담을 수 있습니다. |

따라서 [false, false, false, true, false, true, true, false, true, true]를 리턴합니다.

입력2

infos = ["kim password", "lee abc"]

actions = [

"LOGIN lee abc",

"LOGIN kim password"

]

출력2(리턴 값)

[true, false]

설명2

| **행동** | **결과** | **설명** |
| --- | --- | --- |
| LOGIN lee abc | 허용 | 아이디와 비밀번호가 일치하므로 로그인할 수 있습니다. |
| LOGIN kim password | 거부 | 이미 lee 계정으로 로그인했으므로 거부합니다. |

따라서 [true, false]를 리턴합니다.

족발만 먹는 준과 준의 애인은 족발 레시피를 알고 있으며 냉장고에 남은 재료를 늘 파악하고 있습니다. 준은 재료가 부족하면 장을 보러 갑니다.

레시피: 돼지 족 4kg, 양파 50g, 대파 10cm, 마늘 10g, 고추 4g

상품 가격: 돼지 족(10kg당) 10000원, 양파(100g) 3000원, 대파(30cm) 1000원, 마늘(50g) 2000원, 고추(10g) 1000원

현재 냉장고 상태: 돼지 족 5kg, 양파 100g, 대파 10cm, 마늘 5g, 고추 2g

프로그래머 준은 자신이 먹는 이력에 따른 지출 이력을 확인하고자 합니다. 가령, 1인분, 2인분, 1.5인분 순으로 족발을 만들 경우 1.0, 2.0, 1.5를 입력합니다.

각 상황별로 예상되는 지출 금액을 계산하여 배열 형식으로 return 하도록 solution 메서드를 완성해주세요.  
(결과는 소수점을 버리며, 제한사항을 벗어나는 경우 -1을 return 합니다.)

제한사항

* 준은 1, 2인분을 먹을 수 있으며, 굶지 않습니다.
* 준의 애인은 0, 0.5인분을 먹을 수 있습니다.
* 준의 애인은 매운 음식을 잘 먹지 못하기 때문에, 애인이 먹을 경우 고추를 절반만 넣습니다.
* 상품은 제시된 단위별로만 구매 가능합니다.

입출력 예

| **history** | **result** |
| --- | --- |
| [1.0, 2.0, 1.5] | [3000, 14000, 15000] |
| [1.0, 2.0, 0.0, 1.0] | [-1] |

입출력 예 설명

입출력 예 #1  
아래와 같습니다.

| **번째** | **돼지 족(kg)** | **양파(g)** | **대파(cm)** | **마늘(g)** | **고추(g)** | **구입항목** | **구입비용(원)** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 처음 | 5 | 100 | 10 | 5 | 2 |  |  |
| 1 | 1 | 50 | 0 | 45 | 8 | 마늘 1개, 고추 1개 | 3000 |
| 2 | 3 | 50 | 10 | 25 | 0 | 돼지 1개, 양파 1개, 대파 1개 | 14000 |
| 3 | 7 | 75 | 25 | 10 | 7 | 돼지 1개, 양파 1개, 대파 1개, 고추 1개 | 15000 |

입출력 예 #2  
제한 사항을 벗어나므로 -1을 return 합니다.

우아한테크코스에서는 교육생(이하 크루) 간 소통 시 닉네임을 사용합니다. 간혹 비슷한 닉네임을 정하는 경우가 있는데, 이러할 경우 소통할 때 혼란을 불러일으킬 수 있습니다.



혼란을 막기 위해 크루들의 닉네임 중 **같은 글자가 연속적으로 포함** 될 경우 해당 닉네임 사용을 제한하려 합니다. 이를 위해 같은 글자가 연속적으로 포함되는 닉네임을 신청한 크루들에게 알려주는 시스템을 만들려고 합니다.

신청받은 닉네임 중 **같은 글자가 연속적으로 포함** 되는 닉네임을 작성한 지원자의 이메일 목록을 return 하도록 solution 메서드를 완성해주세요.

제한사항

* 두 글자 이상의 문자가 연속적으로 순서에 맞추어 포함되어 있는 경우 중복으로 간주합니다.
* 크루는 1명 이상 10,000명 이하입니다.
* 이메일은 이메일 형식에 부합하며, 전체 길이는 11자 이상 20자 미만입니다.
* 신청할 수 있는 이메일은 email.com 도메인으로만 주어집니다.
* 닉네임은 한글만 가능하고 전체 길이는 1자 이상 20자 미만입니다.
* result는 이메일에 해당하는 부분의 문자열을 오름차순으로 정렬하고 중복은 제거합니다.

입출력 예

| **forms** | **result** |
| --- | --- |
| [ [jm@email.com, 제이엠], [jason@email.com, 제이슨], [woniee@email.com, 워니], [mj@email.com, 엠제이], [nowm@email.com, 이제엠] ] | [jason@email.com, jm@email.com, mj@email.com] |

입출력 예 설명

forms에는 5명의 크루가 신청서를 작성하였고 이 중 jason@email.com과 jm@email.com, mj@email.com 크루가 중복 닉네임 대상으로 뽑혔습니다. 따라서 이 세 크루의 이메일을 출력합니다.

레벨 2의 팀 프로젝트 미션으로 SNS(Social Networking Service)를 만들고자 하는 팀이 있었습니다. 팀에 속한 크루 중 평소 알고리즘에 관심이 많은 미스터코는 친구 추천 알고리즘을 구현하고자 아래와 같은 규칙을 세웠습니다.

사용자와 함께 아는 친구의 수 = 10점

사용자의 타임 라인에 방문한 횟수 = 1점

사용자 아이디 user와 친구 관계를 담은 이차원 배열 friends, 사용자 타임 라인 방문 기록 visitors가 매개변수로 주어질 때, 미스터코의 친구 추천 규칙에 따라 점수가 가장 높은 순으로 정렬하여 최대 5명을 return 하도록 solution 메서드를 완성해주세요. 이때 추천 점수가 0점인 경우 추천하지 않으며, 추천 점수가 같은 경우는 이름순으로 정렬합니다.

제한사항

* user는 길이가 1 이상 30 이하인 문자열입니다.
* friends는 길이가 1 이상 10,000 이하인 배열입니다.
* friends의 각 원소는 길이가 2인 배열로 [아이디 A, 아이디 B] 순으로 들어있습니다.
  + A와 B는 친구라는 뜻입니다.
  + 아이디는 길이가 1 이상 30 이하인 문자열입니다.
* visitors는 길이가 0 이상 10,000 이하인 배열입니다.
* 사용자 아이디는 알파벳 소문자로만 이루어져 있습니다.
* 동일한 친구 관계가 중복해서 주어지지 않습니다.
* 추천할 친구가 없는 경우는 주어지지 않습니다.

입출력 예

| **user** | **friends** | **visitors** | **result** |
| --- | --- | --- | --- |
| mrko | [ [donut, andole], [donut, jun], [donut, mrko], [shakevan, andole], [shakevan, jun], [shakevan, mrko] ] | [bedi, bedi, donut, bedi, shakevan] | [andole, jun, bedi] |

입출력 예 설명

입출력 예 #1  
mrko는 donut, shakevan과 친구입니다. 두 사람을 제외한 각 사용자들의 추천 점수는 아래와 같습니다.

| **사용자** | **추천 점수** |
| --- | --- |
| andole | 20 |
| jun | 20 |
| bedi | 3 |

위 세 사람을 점수가 가장 높은 순으로 정렬하고, 점수가 같은 andole과 jun은 이름순으로 오름차순 하였을 때, andole, jun, bedi 순입니다.