일상에서 괄호는 순서를 표현하거나, 부가 정보를 제공하거나, 특정 내용을 부각하는 등의 용도로 사용됩니다.  
괄호는 여는 괄호와 닫는 괄호가 한 묶음을 이루어야 하는 특징을 가지고 있으며 다양한 괄호가 존재합니다.  
아래의 규칙에 맞추어 임의의 문자열에 대해 다양한 괄호가 올바르게 사용되었는지 확인할 수 있는 해결법을 제시해 봅시다.

제한 사항

* 임의의 문자열 inputString이 입력됩니다
  + 공백으로만 이루어진 문자열은 입력되지 않습니다
  + 입력 문자열의 길이는 1 이상입니다
  + 최대 길이는 명시되지 않습니다
* 괄호들이 정상적으로 사용되었는지 검증한 결과를 반환합니다
  + 괄호는 아래와 같이 네 종류가 있다고 가정합니다
  + ( ), { }, [ ], < >
  + 괄호의 여닫는 특징만을 검증하며, 열려 있는 어떤 괄호에 대해서 짝이 다른 닫히는 괄호에 영향을 주지 않습니다
  + 예를 들어 ([])와 ([)] 모두 잘 열고 잘 닫았기 때문에 정상입니다
  + 정상이라면 총 괄호 쌍의 수, 비정상(괄호 쌍이 정상적인 순서로 표현되지 않은 경우)이라면 -1을 반환합니다

##### 입출력 예

| **inputString** | **result** |
| --- | --- |
| Hello, world! | 0 |
| line [plus] | 1 |
| if (Count of eggs is 4.) {Buy milk.} | 2 |
| >\_< | -1 |

###### 문제 설명

객관식 5지선다로 치러진 시험의 답안지가 있습니다. 이때, 어떤 두 응시자가 제출한 답안지에 대해서, 부정행위 가능성 지수는 아래와 같은 방식으로 계산할 수 있습니다.

* 의심 문항 = 같은 선택지를 골랐으나 오답인 문항
* 부정행위 가능성 지수 = 총 **의심 문항**의 수 + (가장 긴 연속된 **의심 문항**의 수)2

응시자들이 제출한 답안지들 사이에서 가장 높은 부정행위 가능성 지수를 찾아주세요.

##### 제한 사항

* answer\_sheet는 정답지를 나타냅니다.
  + answer\_sheet는 길이가 1 이상 100 이하인 문자열입니다.
    - 1번 문항의 정답부터 차례대로 저장하고 있습니다.
    - answer\_sheets는 1,2,3,4,5로만 이루어진 문자열입니다.
* sheets는 응시자들이 제출한 답안지입니다.
  + sheets는 원소의 개수가 2 이상 200 이하인 문자열 배열입니다.
    - sheets의 각 원소는 한 응시자가 제출한 답안을 저장하고 있는 문자열입니다.
    - sheets의 모든 원소의 길이는 answer\_sheet의 길이와 같습니다.
    - 즉, 응시자 수(=sheets의 원소 개수)는 2명 이상 200명 이하이고, 문항의 수(=sheets의 각 원소의 길이=answer\_sheet의 길이)는 1개 이상 100개 이하입니다.
    - sheets의 모든 원소는 1,2,3,4,5로만 이루어진 문자열입니다.
* 모든 답안지들 사이에서 가장 높은 부정행위 가능성 지수를 찾아서 return 해주세요.

입출력 예

| **answer\_sheet** | **sheets** | **result** |
| --- | --- | --- |
| "4132315142" | ["3241523133","4121314445","3243523133","4433325251","2412313253"] | 17 |
| "53241" | ["53241", "42133", "53241", "14354"] | 0 |
| "24551" | ["24553", "24553", "24553", "24553"] | 2 |

입출력 예 설명

입출력 예 #1

| **정답** | **4** | **1** | **3** | **2** | **3** | **1** | **5** | **1** | **4** | **2** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 답안 1 | **3** | **2** | **4** | 1 | **5** | **2** | **3** | 1 | **3** | **3** |
| 답안 2 | 4 | 1 | 2 | 1 | 3 | 1 | 4 | 4 | 4 | 5 |
| 답안 3 | **3** | **2** | **4** | 3 | **5** | **2** | **3** | 1 | **3** | **3** |
| 답안 4 | 4 | 4 | 3 | 3 | 3 | 2 | 5 | 2 | 5 | 1 |
| 답안 5 | 2 | 4 | 1 | 2 | 3 | 1 | 3 | 2 | 5 | 3 |

위의 예시는 10개의 객관식 문항에 대해서, 5명의 응시자가 제출한 답안들입니다.  
답안 1, 3 사이에서는 의심 문항이 8개 발견되었고, 가장 긴 연속된 의심 문항의 수는 3입니다. 따라서 답안 1, 3 사이의 부정행위 가능성 지수는 8 + 32 = 17입니다. 길이가 3인 연속된 의심 문항 구간이 두 곳 있지만(1~3번 문항, 5~7번 문항) 한 번만 더해주는 것에 유의해야 합니다.  
답안 4, 5 사이에서는 의심 문항이 3개 발견되었고, 가장 긴 연속된 의심 문항의 수는 2입니다. 따라서 답안 4, 5 사이의 부정행위 가능성 지수는 3+22 = 7입니다.

입출력 예 #2

| **정답** | **5** | **3** | **2** | **4** | **1** |
| --- | --- | --- | --- | --- | --- |
| 답안 1 | 5 | 3 | 2 | 4 | 1 |
| 답안 2 | 4 | 2 | 1 | 3 | 3 |
| 답안 3 | 5 | 3 | 2 | 4 | 1 |
| 답안 4 | 1 | 4 | 3 | 5 | 4 |

어느 답안들 사이에도 의심 문항이 존재하지 않습니다.  
답안 1, 3이 모두 같지만 정답과 일치하므로, 의심 문항으로 볼 수는 없습니다.  
따라서, 0( = 0+02 )을 return 해주어야 합니다.

입출력 예 #3

| **정답** | **2** | **4** | **5** | **5** | **1** |
| --- | --- | --- | --- | --- | --- |
| 답안 1 | 2 | 4 | 5 | 5 | **3** |
| 답안 2 | 2 | 4 | 5 | 5 | **3** |
| 답안 3 | 2 | 4 | 5 | 5 | **3** |
| 답안 4 | 2 | 4 | 5 | 5 | **3** |

어느 두 개의 답안을 골라서 비교하더라도, 5번 문항이 유일한 의심 문항입니다.  
따라서, 2( = 1+12 )을 return 해주어야 합니다.

###### 문제 설명

군데군데 손상된 도로가 있습니다. 이 도로를 적절하게 보수하여, 자동차 경주 대회를 열려고 합니다. 자동차 경주 대회가 열리려면, 손상되지 않은 최대한 긴 구간이 필요합니다. 아래는 도로의 손상된 부분을 0, 정상적인 부분을 1로 나타낸 예시입니다.

#### 111011110011111011111100011111

만약 도로의 손상된 부분을 최대 3곳까지만 보수할 수 있는 재료가 있다면, 아래와 같이 도로를 보수하여 길이가 18인 정상 도로 구간을 만들 수 있습니다.

#### 1110(111111111111111111)00011111

도로를 보수해서 만들 수 있는 가장 긴 정상 도로 구간의 길이를 구해주세요.

##### 제한 사항

* road는 도로의 상태를 나타냅니다.
  + road는 길이 1 이상 300,000 이하인 문자열입니다.
    - road는 0과 1로만 이루어져 있습니다.
    - 도로의 손상된 부분을 0, 정상 부분은 1로 나타냅니다.
* n은 보수 가능한 최대 횟수를 의미합니다.
  + n은 0 이상 300,000 이하인 자연수입니다.
* 만들 수 있는 가장 긴 정상 도로 구간의 길이를 return 하면 됩니다.

##### 입출력 예

| **road** | **n** | **result** |
| --- | --- | --- |
| "111011110011111011111100011111" | 3 | 18 |
| "001100" | 5 | 6 |

###### 문제 설명

슬프게도 Cony Bank에서 장애가 발생하여 데이터베이스에서 데이터가 삭제되었습니다.  
Cony Bank의 기술 담당자 브라운은 스냅숏(snapshot) 데이터와 트랜잭션 로그(transaction log)를 사용하여 데이터베이스를 복구하려 합니다.

##### 제한 사항

[[계좌 이름, 잔액], ... ],

* 스냅숏 데이터 snapshots는 계좌 이름과 잔액이 문자열(String)의 배열로 입력됩니다
  + 계좌 이름은 숫자와 대소문자가 구분되는 영문자로 구성된 최대 10자리의 문자열입니다
  + 스냅숏 데이터의 개수는 1 이상 105 이하입니다
  + 모든 계좌는 서로 계좌 이름이 다릅니다
  + 잔액은 100,000보다 작고, 0보다 크거나 같은 숫자의 문자열입니다

[ID2, WITHDRAW, 계좌 이름2, 출금 금액2], ... ]

* 트랜잭션 로그 transactions는 ID, 종류, 계좌 이름, 출금 금액이 문자열(String)의 배열로 입력됩니다
  + 트랙잭션 로그의 개수는 1 이상 105 이하입니다
  + 모든 트랙잭션 로그에는 ID가 있습니다
  + ID는 모든 트랜잭션 로그를 각각 구분할 수 있는 고유 ID입니다
    - 100,000보다 작고 0보다 큰 정수의 문자열입니다
    - 트랜잭션 로그는 누락이 없도록 모든 데이터베이스에서 읽어 왔기 때문에 중복이 발생할 수 있고, ID 순서가 랜덤입니다
  + 같은 트랜잭션이 중복 적용되지 않도록, ID를 확인하여 한 번만 적용되도록 해야 합니다
    - 단, ID가 동일한 경우 트랜잭션 로그의 내용도 동일하다
  + 스냅숏 데이터 이후에 추가된 계좌가 있을 수 있으므로, 스냅숏 데이터에서 존재하지 않았던 계좌가 사용될 수 있습니다
  + 트랜잭션의 종류는 SAVE와 WITHDRAW 2가지가 있습니다
  + SAVE: 입금에 대한 정보로, 계좌 이름과 입금 금액 정보가 문자열 배열로 함께 입력됩니다
    - 입금 금액은 0 이상의 정수 문자열입니다
  + WITHDRAW: 출금에 대한 정보로, 계좌 이름과 출금 금액 정보가 문자열 배열로 함께 입력됩니다
    - 출금 금액은 0 이상의 정수 문자열입니다
  + 잔액이 음수가 되는 트랜잭션은 없습니다

[계좌 이름, 잔액]

* 계좌의 이름과 잔액을 문자열 배열로 반환합니다
  + 계좌의 이름을 기준으로 오름차순 정렬하여 반환합니다
  + 잔액은 쉼표 없이 숫자로만 이루어진 문자열로 출력합니다
  + 잔액은 100,000보다 작고, 0보다 크거나 같은 숫자의 문자열입니다

##### 입출력 예

##### snapshots

[

["ACCOUNT1", "100"],

["ACCOUNT2", "150"]

]

##### transactions

[

["1", "SAVE", "ACCOUNT2", "100"],

["2", "WITHDRAW", "ACCOUNT1", "50"],

["1", "SAVE", "ACCOUNT2", "100"],

["4", "SAVE", "ACCOUNT3", "500"],

["3", "WITHDRAW", "ACCOUNT2", "30"]

]

##### result

[

["ACCOUNT1", "50"],

["ACCOUNT2", "220"],

["ACCOUNT3", "500"]

]

###### 문제 설명

신입 사원 브라운은 문서 시스템을 다루는 팀에서 일하게 되었습니다.  
기존의 문서 시스템은 문서의 특징을 잘 나타낼 수 있도록 관련 키워드를 등록하여 관리하는 태그 기능을 포함하고 있습니다.

브라운은 기존 문서 정보를 기반으로 복수의 태그를 이용하여 관련이 있는 문서의 목록을 반환하는 새로운 기능을 만들게 되었습니다.  
아래의 제약 조건을 확인하고 새 검색 기능을 구현해 주세요.

##### 제한 사항

[

["문서1", "태그1", "태그2", "태그3", ...],

["문서2", "태그2", "태그3", "태그5", ...],

["문서3", "태그99", "태그100", ..],

...

]

* dataSource는 2차원 배열로 여러 건의 문서와 그에 따른 태그 정보를 포함합니다
  + 한 문서에 대한 정보는 배열로 구성되어 있으며, 첫 번째 원소는 문서 이름을, 두 번째 원소부터는 태그를 나타냅니다
  + 태그의 수는 문서마다 다를 수 있습니다
  + 한 문서 정보 안에 중복된 태그는 없습니다
  + 서로 다른 문서가 같은 태그를 가질 수 있습니다
  + 문서의 개수는 1 이상 105 이하입니다
  + 태그의 개수는 0 이상 105 이하입니다

["태그1", "태그2", "태그3", "태그5"]

* tags는 검색에 사용될 태그 문자열의 배열입니다
  + 태그의 개수는 1 이상 104 이하입니다

["문서1", "문서2", ...]

* 주어진 태그 중 하나 이상 동일한 태그를 가지는 문서들을 반환합니다
  + 주어진 태그를 가장 많이 포함하는 문서일수록 앞에 위치시킵니다
  + 포함하는 태그의 수가 동일할 때는 문서 이름을 사전식 순서(lexicographical order)로 정렬합니다
  + 페이지네이션(pagination) 기능을 적용하기 위해서 검색 결과는 정렬된 문서들 중 상위 10건만 반환합니다

##### 입출력 예

##### dataSource

[

["doc1", "t1", "t2", "t3"],

["doc2", "t0", "t2", "t3"],

["doc3", "t1", "t6", "t7"],

["doc4", "t1", "t2", "t4"],

["doc5", "t6", "t100", "t8"]

]

##### tags

["t1", "t2", "t3"]

##### result

["doc1", "doc2", "doc4", "doc3"]

###### 문제 설명

리눅스의 쉘 명령어가 하는 일을 유사하게 따라 하는 프로그램을 작성하려고 합니다.

다음 세 가지 명령어가 동작하도록 프로그램을 작성해주세요.

### 1.

| **이름** | **mkdir** |
| --- | --- |
| 사용 방법 | mkdir directory |
| 설명 | 특정 위치에 디렉토리를 생성합니다. |

### 2.

| **이름** | **rm** |
| --- | --- |
| 사용 방법 | rm directory |
| 설명 | 특정 디렉토리를 삭제합니다. 삭제한 디렉토리의 하위 디렉토리들 또한 함께 삭제됩니다. |

### 3.

| **이름** | **cp** |
| --- | --- |
| 사용 방법 | cp source dest |
| 설명 | source 디렉토리를 dest로 복사합니다. source 디렉토리의 하위 디렉토리들 또한 복사되어야 합니다. |

위 명령을 수행할 시스템에는 오직 디렉토리만 존재하며, 최상위 디렉토리는 /로 표시하고 그 하위 디렉토리는 /root, /usr, /tmp, /tmp/tmp2와 같이 구분합니다.

##### 제한사항

* directory는 시스템의 디렉토리 구조를 나타내는 문자열 배열입니다.
  + directory의 길이는 1 이상 20 이하입니다.
  + directory의 원소는 각 디렉토리의 절대 경로(최상위 디렉토리부터의 경로)를 나타냅니다.
    - 시스템상의 모든 디렉토리에 대한 경로가 주어집니다.
    - 각 디렉토리 이름은 길이가 1 이상 10 이하입니다.
    - 경로를 나타내는 문자열은 알파벳 소문자와 '/' 로만 이루어져 있습니다.
    - 한 디렉토리 안에 같은 이름의 디렉토리가 두 개 이상 있는 경우는 없으며, 또한 잘못된 경로가 주어지는 경우는 없습니다.
    - 디렉토리 경로는 사전 순으로 정렬되어 주어집니다.
* command는 명령어들이 담긴 문자열 배열입니다.
  + command의 길이는 1 이상 20 이하입니다.
  + command의 원소는 쉘 명령어를 나타내는 문자열입니다.
    - 문제에 주어진 명령어만 입력으로 주어지며, 잘못된 경로가 주어지는 경우는 없습니다.
    - 경로는 항상 절대 경로(최상위 디렉토리부터의 경로)로만 주어집니다.
    - cp명령의 경우 자기 자신의 하위 디렉토리에 복사하는 명령은 주어지지 않습니다.
    - cp명령의 경우 복사하는 위치에 이름이 동일한 다른 디렉토리가 이미 있는 경우는 주어지지 않습니다.
    - rm / 명령은 입력으로 주어지지 않습니다.
    - rm 명령은 항상 디렉토리 하나를 정상적으로 삭제하는 경우만 입력으로 주어집니다. (존재하지 않거나 잘못된 경로, 없는 디렉토리 등을 삭제하는 경우는 없습니다)
    - mkdir 명령은 항상 디렉토리 하나를 정상적으로 생성하는 경우만 입력으로 주어집니다. (존재하지 않거나 잘못된 경로, 해당 경로에 같은 이름의 디렉토리가 이미 있는 경우, 중간 경로에 해당하는 디렉토리가 없는 경우 등은 입력으로 주어지지 않습니다.)
* 모든 명령어를 수행한 후 디렉토리 구조를 문자열 형태로 배열에 담아 사전 순으로 정렬해서 return해주세요.
  + 입력되는 디렉토리 구조와 마찬가지로, 모든 디렉토리에 대한 절대 경로를 문자열 형태로 담은 뒤 사전 순으로 정렬해서 return 하면 됩니다.
  + '/'가 알파벳보다 사전 순으로 더 앞에 온다고 가정합니다.
* 명령어 중 한 가지 또는 두 가지만 구현한 경우에도 부분 점수를 획득할 수 있습니다.

##### 입출력 예 #1

##### directory

[

"/",

"/hello",

"/hello/tmp",

"/root",

"/root/abcd",

"/root/abcd/etc",

"/root/abcd/hello"

]

##### command

[

"mkdir /root/tmp",

"cp /hello /root/tmp",

"rm /hello"

]

##### result

[

"/",

"/root",

"/root/abcd",

"/root/abcd/etc",

"/root/abcd/hello",

"/root/tmp",

"/root/tmp/hello",

"/root/tmp/hello/tmp"

]

##### 입출력 예 #2

##### directory

[

"/"

]

##### command

[

"mkdir /a",

"mkdir /a/b",

"mkdir /a/b/c",

"cp /a/b /",

"rm /a/b/c"

]

##### result

[

"/",

"/a",

"/a/b",

"/b",

"/b/c"

]

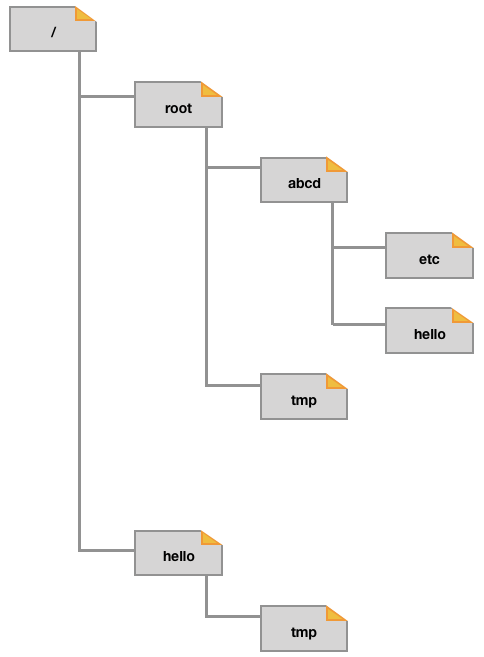
##### 입출력 예 설명

입력으로 주어진 초기 디렉토리 구성은 다음과 같습니다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

* "mkdir /root/tmp" 명령을 수행한 후 디렉토리 구성은 다음과 같습니다.

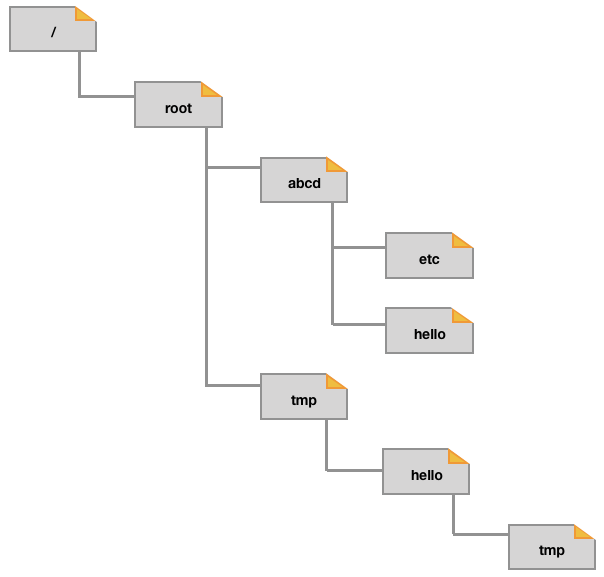


* "cp /hello /root/tmp" 명령을 수행한 후 디렉토리 구성은 다음과 같습니다.

지도이(가) 표시된 사진

자동 생성된 설명

* "rm /hello" 명령을 수행한 후 최종 디렉토리 구성은 다음과 같습니다.



입출력 예 #2

명령을 모두 처리한 후 디렉토리 구성은 다음과 같습니다.

스크린샷이(가) 표시된 사진

자동 생성된 설명