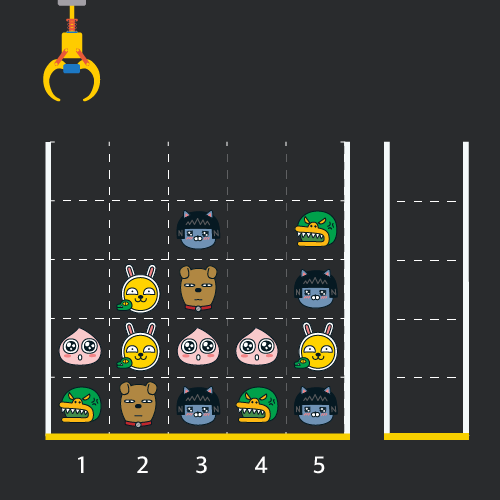
**크레인 인형뽑기 게임**



**[문제]**

사용자는 크레인을 좌우로 움직여서 가장 위에 있는 인형을 집어 올릴 수 있다.

집어 올린 인형은 바구니에 쌓이게 되는데 같은 모양의 인형 두 개가 바구니에 연속해서 쌓이게 되면 두 인형은 바구니에서 사라지게 된다. (바구니에는 모든 인형이 들어갈 수 있음)

인형이 없는 곳에서 크레인을 작동시키면 아무런 일도 일어나지 않는다.

격자의 상태가 담긴 2차원 배열 board와 크레인을 작동시킨 위치가 담긴 배열 moves가 매개변수로 주어질 때, 크레인을 모두 작동시킨 후 터트려져 사라진 인형의 개수를 return하는 solution함수를 완성하라.

**[제한사항]**

* board의 크기는 5x5 이상 30x30 이하
* board에 0은 빈칸, 1~100은 각기 다른 인형의 모양을 의미
* moves의 크기는 1~1000이며 각 원소 값은 1 이상 board 가로 크기 이하의 자연수

**[Python]**

|  |  |
| --- | --- |
| def solution(board, moves):      answer = 0      pick = []      index = -1        for i in moves :          for j in range(len(board[0])) :              x = board[j][i - 1]              board[j][i - 1] = 0              if x :                  if not pick :                      pick.append(x)                      index += 1                  elif pick[index] == x :                      answer += 2                      del pick[index]                      index -= 1                  else :                      pick.append(x)                      index += 1                  break        return answer | [cs](http://colorscripter.com/info#e) |

**[MEMO]**

<Python>

* range() 빼먹지 말기

**완주하지 못한 선수**

**[문제]**

단 한 명의 선수를 제외하고는 모든 선수가 마라톤을 완주하였다.

참여한 선수들의 이름이 담긴 배열 participant와 완주한 선수들의 이름이 담긴 배열 completion이 주어질 때, 완주하지 못한 선수의 이름을 return하는 solution 함수를 작성하라.

**[제한사항]**

* 선수는 1~100000명 이하
* completion의 길이는 participant의 길이보다 1 작음
* 참가자의 이름은 1~20개의 알파벳 소문자로 이루어짐
* 동명이인이 있을 수 있음

**[Python]**

|  |  |
| --- | --- |
| def solution(participant, completion):      answer = ''      participant.sort()      completion.sort()      i = 0      while i < len(completion) :          if participant[i] != completion[i] :              answer = participant[i]              break          i += 1      if i == len(completion) :          answer = participant[i]      return answer | [cs](http://colorscripter.com/info#e) |

**[MEMO]**

<Python>

* collections 쓰면 더 간단함
* hash() 쓰는 방법도 있음

**모의고사**

**[문제]**

1번 수포자가 찍는 방식: 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, …

2번 수포자가 찍는 방식: 2, 1, 2, 3, 2, 4, 2, 5, 2, 1, 2, 3, 2, 4, 2, 5, …

3번 수포자가 찍는 방식: 3, 3, 1, 1, 2, 2, 4, 4, 5, 5, 3, 3, 1, 1, 2, 2, 4, 4, 5, 5, …

정답이 순서대로 들은 배열 answers가 주어졌을 때, 가장 많은 문제를 맞힌 사람이 누구인지 배열에 담아 return하도록 solution 함수를 작성하라.

**[제한사항]**

* 시험 문제는 최대 10000문제
* 문제의 정답은 1, 2, 3, 4, 5 중 하나
* 가장 높은 점수를 받은 사람이 여럿일 경우, return하는 값을 오름차순으로 정렬

**[Python]**

|  |  |
| --- | --- |
| def solution(answers):      answer = []      sel = [[1, 2, 3, 4, 5], [2, 1, 2, 3, 2, 4, 2, 5], [3, 3, 1, 1, 2, 2, 4, 4, 5, 5]]      score = [0, 0, 0]      for i in range(len(answers)) :          x = answers[i]          if x == sel[0][i % 5] :              score[0] += 1          if x == sel[1][i % 8] :              score[1] += 1          if x == sel[2][i % 10] :              score[2] += 1      max\_score = max(score)      for i in range(len(score)) :          if max\_score == score[i] :              answer.append(i + 1)      return answer  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

**[MEMO]**

<Python>

* for idx, answer in enumerate(list) 하면 idx에는 반복횟수가 들어가고 answer에는 list값이 들어감

**K번째 수**

**[문제]**

배열 array의 i번째 숫자부터 j번째 숫자까지 자르고 정렬했을 때, k번째에 있는 수를 구하려 한다.

배열 array, [I, j, k]를 원소로 가진 2차원 배열 commands가 매개변수로 주어질 때, commands의 모든 원소에 대해 앞서 설명한 연산을 적용했을 때 나온 결과를 배열에 담아 return 하도록 solution 함수를 작성하라.

**[제한사항]**

* array 길이는 1 이상 100 이하
* array 각 원소는 1 이상 100 이하
* commands의 길이는 1 이상 50 이하
* commands의 각 원소의 길이는 3

**[Python]**

|  |  |
| --- | --- |
| def solution(array, commands):      answer = []      temp = []      for i in range(len(commands)) :          temp = []          for j in range(commands[i][0] - 1, commands[i][1]) :              temp.append(array[j])          temp.sort()          answer.append(temp[commands[i][2] - 1])      return answer  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

**[MEMO]**

<Python>

* 인덱스 변수 항상 조심하기
* 초기화가 필요한 변수가 있는지 한 번 더 확인하기
* list[:]로 쪼갤 수 있음
* return list(map(lamda x : sorted(array[x[0]-1 : x[1]])[x[2]-1], commands))