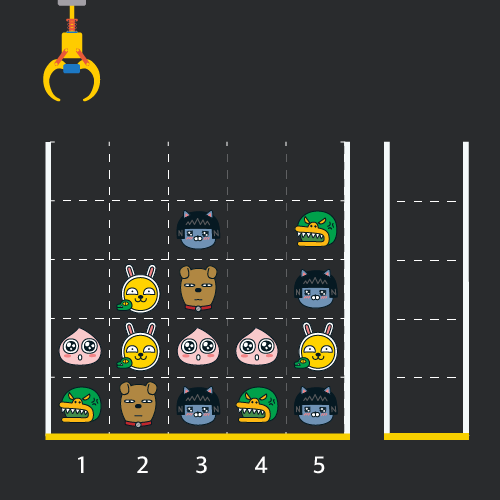
**크레인 인형뽑기 게임**



**[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

**[JAVA]**

|  |  |
| --- | --- |
| import java.util.Stack;  class Solution {     public int solution(int[][] board, int[] moves) {          int answer = 0;          int n = board.length;            Stack<Integer>[] stack = new Stack[n + 1];          for (int i = 0; i <= n; i++) {              stack[i] = new Stack<Integer>();          }            for (int i = n - 1; i >= 0; i--) {              for (int j = 0; j < n; j++) {                  if (board[i][j] != 0) {                      stack[j + 1].add(board[i][j]);                  }              }          }            for (int i = 0; i < moves.length; i++)  {              if (!stack[moves[i]].isEmpty()) {                  int pick = stack[moves[i]].pop();                  if (!stack[0].isEmpty() && stack[0].peek() == pick) {                      answer += 2;                      stack[0].pop();                  } else {                      stack[0].add(pick);                  }              }            }            return answer;      }  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

**[MEMO]**

<JAVA>

* 횟수와 개수는 다름에 주의!!
* 이중 for문에서 인덱스 사용 항상 조심하기
* 반복문 최대한 줄이기 위해 한 번에 처리할 수 없는지 고민해볼 것

**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

**[JAVA]**

|  |  |
| --- | --- |
| import java.util.\*;  class Solution {    public String solution(String[] participant, String[] completion) {          Arrays.sort(participant);          Arrays.sort(completion);            String answer = "";          int i = 0;          for(i = 0; i < completion.length; i++) {              if(!participant[i].equals(completion[i])) {                  answer = participant[i];                  return answer;              }          }          answer = participant[i];          return answer;      }  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

**[MEMO]**

<JAVA>

* 배열 정렬은 Arrays.sort() (import.java.util.Arrays 필요)
* String끼리 비교는 .equals()로
* 답 찾으면 바로 return 해주기

**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

**[JAVA]**

|  |  |
| --- | --- |
| import java.util.Arrays;  class Solution {   public static int[] solution(int[] answers) {          int[] answer = {};          int[] count = new int[3];          int[] sel = {3, 1, 2, 4, 5};          int max = 0;            for (int i = 0; i < answers.length; i++) {              int x = answers[i];                if (x == i % 5 + 1) {                  count[0]++;                  if (max < count[0]) max = count[0];              }              if (i % 2 == 0) {if (x == 2) {                  count[1]++;                  if (max < count[1]) max = count[1];                  }              }              else if (x == i - (8 \* (i / 8) + (i % 8 / 4) + (i % 8 / 7))) {                  count[2]++;                  if (max < count[2]) max = count[2];              }              if (x == sel[i / 2 % 5]) {                  count[2]++;                  if (max < count[2]) max = count[2];              }          }            int count1 = 0;          for (int i = 0; i < count.length; i++) {              if (max == count[i]) {                  count[count1] = i + 1;                  count1++;              }          }          answer = Arrays.copyOf(count, count1);            return answer;      }  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

**[MEMO]**

<JAVA>

* Math.max(Math.max())로도 3개 중 제일 큰 값 구할 수 있음
* 인덱스 변수 항상 주의하기!!

**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

**[JAVA]**

|  |  |
| --- | --- |
| import java.util.Arrays;  class Solution {      public int[] solution(int[] array, int[][] commands) {          int[] answer = new int[commands.length];          int[] temp = {};            for(int i = 0; i < commands.length; i++) {                temp = Arrays.copyOfRange(array, commands[i][0] - 1, commands[i][1]);              Arrays.sort(temp);              answer[i] = temp[commands[i][2] - 1];            }          return answer;      }  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

**[MEMO]**

<JAVA>

* 2차원 배열이라고 무조건 이중 반복문 쓰는 거 아님
* 인덱스 범위 주의!!