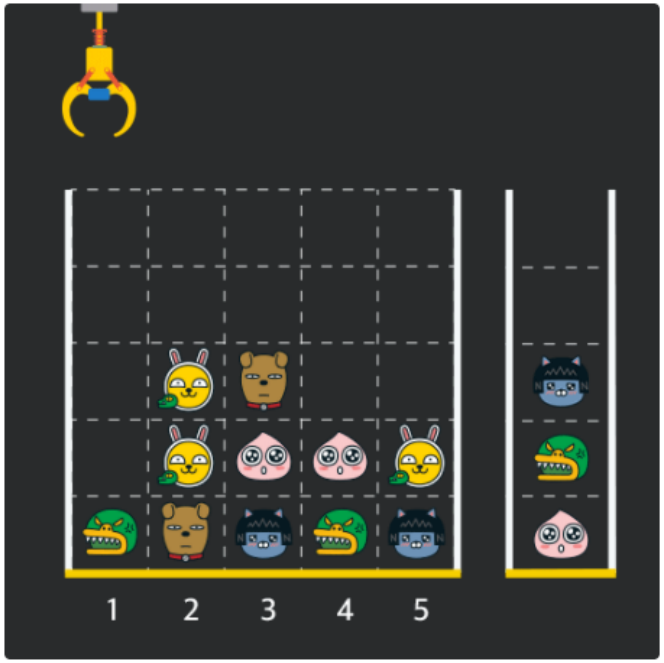
프로그래머스 코딩테스트 기록

이우람

크레인 인형뽑기 게임 (C++)



크레인 작동 시 인형이 집어지지 않는 경우는 없으나 만약 인형이 없는 곳에서 크레인을 작동시키는 경우에는 아무런 일도 일어나지 않습니다. 또한 바구니는 모든 인형이 들어갈 수 있을 만큼 충분히 크다고 가정합니다. (그림에서는 화면표시 제약으로 5칸만으로 표현하였음)

게임 화면의 격자의 상태가 담긴 2차원 배열 board와 인형을 집기 위해 크레인을 작동시킨 위치가 담긴 배열 moves가 매개변수로 주어질 때, 크레인을 모두 작동시킨 후 터트려져 사라진 인형의 개수를 return 하도록 solution 함수를 완성해주세요.

```cpp
#include <string>
#include <vector>

using namespace std;

int solution(vector<vector<int>> board, vector<int> moves) {
    int answer = 0;
    int save = 0;
    vector<int> result;

    for(int i = 0; i < moves.size(); i++){
        for(int j = 0; j < board.size(); j++){
            if(board[j][moves[i] - 1] != 0){
                save = board[j][moves[i] - 1];
                if(!result.empty() && result.back() == save){
                    result.pop_back();
                    answer += 2;
                }else{
                    result.push_back(save);
                }
                board[j][moves[i] - 1] = 0;
                break;
            }
        }
    }
    return answer;
}
```

[1차] 비밀지도 (C++)

문제 설명

비밀지도

네오는 평소 프로도가 비상금을 숨겨놓는 장소를 알려줄 비밀지도를 손에 넣었다. 그런데 이 비밀지도는 숫자로 암호화되어 있어 위치를 확인하기 위해서는 암호를 해독해야 한다. 다행히 지도 암호를 해독할 방법을 적어놓은 메모도 함께 발견했다.

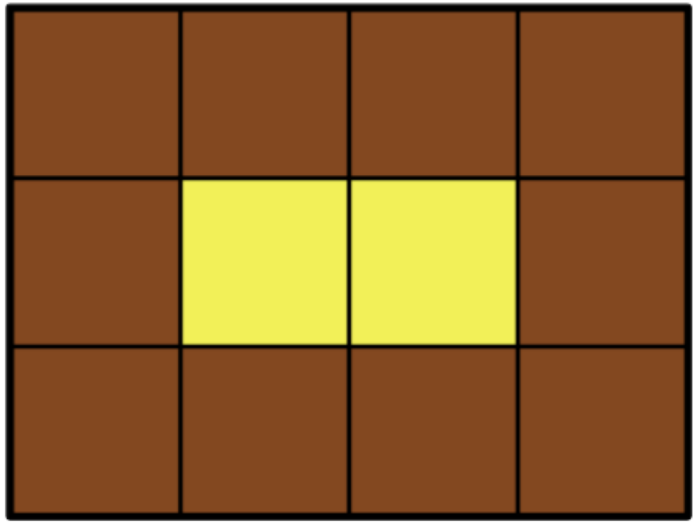
1. 지도는 한 변의 길이가 `n` 인 정사각형 배열 형태로, 각 칸은 "공백"(" ") 또는"벽"("#") 두 종류로 이루어져 있다.
2. 전체 지도는 두 장의 지도를 겹쳐서 얻을 수 있다. 각각 "지도 1"과 "지도 2"라고 하자. 지도 1 또는 지도 2 중 어느 하나라도 벽인 부분은 전체 지도에서도 벽이다. 지도 1과 지도 2에서 모두 공백인 부분은 전체 지도에서도 공백이다.
3. "지도 1"과 "지도 2"는 각각 정수 배열로 암호화되어 있다.
4. 암호화된 배열은 지도의 각 가로줄에서 벽 부분을 `1` , 공백 부분을 `0` 으로 부호화했을 때 얻어지는 이진수에 해당하는 값의 배열이다.

```cpp
#include <string>
#include <vector>

using namespace std;

vector<string> solution(int n, vector<int> arr1, vector<int> arr2) {
    vector<string> answer;
    string a = "";
    for(int i = 0; i < arr1.size(); i++){
        a = "";
        for(int j = 0; j < n; j++){
            if(arr1[i] % 2 == 1 || arr2[i] % 2 == 1){
                a = "#" + a;
            }else {
                a = " " + a;
            }
             arr1[i] = arr1[i] / 2;
            arr2[i] = arr2[i] / 2;
        }
        answer.push_back(a);
    }
    return answer;
}
```

카펫 (Java)

문제 설명

Leo는 카펫을 사러 갔다가 아래 그림과 같이 중앙에는 노란색으로 칠해져 있고 테두리 1줄은 갈색으로 칠해져 있는 격자 모양 카펫을 봤습니다.



Leo는 집으로 돌아와서 아까 본 카펫의 노란색과 갈색으로 색칠된 격자의 개수는 기억했지만, 전체 카펫의 크기는 기억하지 못했습니다.

Leo가 본 카펫에서 갈색 격자의 수 brown, 노란색 격자의 수 yellow가 매개변수로 주어질 때 카펫의 가로, 세로 크기를 순서대로 배열에 담아 return 하도록 solution 함수를 작성해주세요.

```java
class Solution {
    public int[] solution(int brown, int yellow) {
        int[] answer = new int[2];

        int sum = 0;
        int save = 0;
        for (int i = 1; i <= (brown + yellow); i++) {
            save = (brown + yellow) / i;
            if ((brown + yellow) % i == 0 && save <= i) {
                sum = (i + i) + ((save - 2) * 2);
                if (sum == brown) {
                    answer[0] = i;
                    answer[1] = save;
                    break;
                }
            }
        }
        return answer;
    }
}
```

예상 대진표 (Java)

문제 설명

△△ 게임대회가 개최되었습니다. 이 대회는 N명이 참가하고, 토너먼트 형식으로 진행됩니다. N명의 참가자는 각각 1부터 N번을 차례대로 배정받습니다. 그리고, 1번↔2번, 3번↔4번, ... , N-1번↔N번의 참가자끼리 게임을 진행합니다. 각 게임에서 이긴 사람은 다음 라운드에 진출할 수 있습니다. 이때, 다음 라운드에 진출할 참가자의 번호는 다시 1번부터 N/2번을 차례대로 배정받습니다. 만약 1번↔2번 끼리 겨루는 게임에서 2번이 승리했다면 다음 라운드에서 1번을 부여받고, 3번↔4번에서 겨루는 게임에서 3번이 승리했다면 다음 라운드에서 2번을 부여받게 됩니다. 게임은 최종 한 명이 남을 때까지 진행됩니다.

이때, 처음 라운드에서 A번을 가진 참가자는 경쟁자로 생각하는 B번 참가자와 몇 번째 라운드에서 만나는지 궁금해졌습니다. 게임 참가자 수 N, 참가자 번호 A, 경쟁자 번호 B가 함수 solution의 매개변수로 주어질 때, 처음 라운드에서 A번을 가진 참가자는 경쟁자로 생각하는 B번 참가자와 몇 번째 라운드에서 만나는지 return 하는 solution 함수를 완성해 주세요. **단, A번 참가자와 B번 참가자는 서로 붙게 되기 전까지 항상 이긴다고 가정합니다.**

```java
class Solution{
    public int solution(int n, int a, int b){
        int answer = 0;

        while (n > 0) {
            if (a % 2 == 1) a += 1;
            if (b % 2 == 1) b += 1;

            if (Math.abs(a - b) <= 1) {
                answer++;
                break;
            }

            if (a > 1) a /= 2;
            if (b > 1) b /= 2;

            answer++;
            n = n / 2;
        }
        return answer;
    }
}
```

124 나라의 숫자 (Java)

문제 설명

124 나라가 있습니다. 124 나라에서는 10진법이 아닌 다음과 같은 자신들만의 규칙으로 수를 표현합니다.

1. 124 나라에는 자연수만 존재합니다.
2. 124 나라에는 모든 수를 표현할 때 1, 2, 4만 사용합니다.

예를 들어서 124 나라에서 사용하는 숫자는 다음과 같이 변환됩니다.

10진법	124 나라	10진법	124 나라
1	1	6	14
2	2	7	21
3	4	8	22
4	11	9	24
5	12	10	41

자연수 n이 매개변수로 주어질 때, n을 124 나라에서 사용하는 숫자로 바꾼 값을 return 하도록 solution 함수를 완성해 주세요.

```java
class Solution {
    public String solution(int n) {
        String answer = "";
        int rule = 3;
        int last = 3;

        while (n > 0) {
            rule = 3;
            last = 3;

            if (n % rule == 0) {
                answer = "4" + answer;
                last = (last + (n / rule - 2) + (n / rule));
                n = n - last;
            } else if (n % rule == 1) {
                answer = "1" + answer;
                n = (n - (n % rule)) / rule;
            } else if (n % rule == 2) {
                answer = "2" + answer;
                n = (n - (n % rule)) / rule;
            }
        }
        return answer;
    }
}
```

피보나치 수 (Java)

문제 설명

피보나치 수는 F(0) = 0, F(1) = 1일 때, 1 이상의 n에 대하여 F(n) = F(n-1) + F(n-2) 가 적용되는 수 입니다.

예를들어

- F(2) = F(0) + F(1) = 0 + 1 = 1
- F(3) = F(1) + F(2) = 1 + 1 = 2
- F(4) = F(2) + F(3) = 1 + 2 = 3
- F(5) = F(3) + F(4) = 2 + 3 = 5

와 같이 이어집니다.

2 이상의 n이 입력되었을 때, n번째 피보나치 수를 1234567으로 나눈 나머지를 리턴하는 함수, solution을 완성해 주세요.

```java
class Solution {
    public int solution(int n) {
        int div = 1234567;
        long last = 1;// 시작
        long next = 1;// 다음
        long pbnc = 0;// 결과

        if (n == 0) {
            pbnc = 0;
        } else if (n == 1 || n == 2) {
            pbnc = 1;
        } else {
            for (int i = 0; i < n - 2; i++) {
                pbnc = (last % div) + (next % div);
                last = next % div;
                next = pbnc % div;
            }
        }
        int answer = (int) pbnc % div;
        return answer;
    }
}
```

소수 찾기 (Java)

문제 설명

1부터 입력받은 숫자 n 사이에 있는 소수의 개수를 반환하는 함수, solution을 만들어 보세요.

소수는 1과 자기 자신으로만 나누어지는 수를 의미합니다.
(1은 소수가 아닙니다.)

```java
class Solution {
    public int solution(int n) {
        int[] prime = new int[(int) n + 1];
        int answer = 0;
        prime[0] = 1;
        prime[1] = 1;

        for (int i = 2; i <= n; i++) {
            if (prime[i] != 1) {
                for (int j = i + i; j <= n; j = j + i) {
                    prime[j] = 1;
                }
            }
        }

        for (int i = 0; i <= n; i++) {
            if (prime[i] == 0) {
                answer++;
            }
        }
        return answer;
    }
}
```