Problem Analysis

Friendly URL

This is a basic string manipulation problem. No complex algorithm or technique is required. Just take the input process the string to create the converted friendly URL. This is a pure coding problem.

Get the Numbers

This is also an easy problem. You just need to take input and process it to generate the output. No algorithm or tick for this problem. Only thing needed is accurate coding. This is also a pure coding problem.

Kick the Football

Projectile motion equation:

y = bx - cx^2

Where

b = tan(angle)

c = g/2(iv * cos(angle))^2

The unit of angle is degree.

Calculate the value of b and c for given angle and initialVelocity, iv.

Sort all players corresponding to x axis.

Collusion detection:

For each player set the player x coordinate value to the equation and get the y.

If the value of y is between y1 and y2 of player then the collusion occurs and player receives the ball. Print the player name.

Mina Raju

The problem is based on minmax algorithm. In minmax algorithm, one player from a state always tries to reach a state which is the other player losing state. If a player can't find such state, than the current state is a losing state of the player. In the problem description, meena has n and raju has m marbles initially. Meena always starts first. Now we calculate for all states that which state is a winning or losing

state for Meena and Raju. Let R = 0. It is a losing state for both Meena and Raju. But according to problem description, 1st move comes from meena and meena can't able to make any move or take any marble from this state. So, meena will lose. Now from all states 1 to R meena or Raju always try to find a valid state which is losing state for the opponent. If find than the state is a wining sate, if not, then it is a losing state. We use a dynamic programming technique here. We have 2 dimensional table dp[][], which stores the result for all states according to meena and raju .

```
for(int i=1; i<=R; i++)
{
    for(int j=0;j<=1;j++)  // let 0 is meena and 1 is raju .
    {
        dp[i][j] = false ;// Initially losing state for a player .
        for(int k=0; k<V[j].size(); k++)
        {
            if(i-V[j][k]>=0 && dp[i-V[j][k]][!j]==false) dp[i][j] = true ;
            // If find any state that is opponents losing state, than the
state is winning state for the player
        }
    }
}
```

Time Complexity O(R*2*(n+m)) .
For better understand of minmax algorithm please visit: http://goo.gl/FsYKis

Pagination

This is another simple adhoc problem. Just calculate the boundary value carefully and print all numbers without dots if page number is less equal than 6.

Product Owner's Problem

The problem can be solved using combinatorics, the constraints and the time limit of the problem allow a simple O (n^3) dynamic programming solution to be accepted. DP recurrence is straight-forward.

Let,

DP[x][y] denote the number of ways of completing y tasks by the first x programmers. So, our answer will be DP[m][n]. The DP recurrence relation is as follows:

$$DP[i][j] = \Sigma DP[i-1][j-k] \text{ where } X[i] <= k <= Y[i]$$

As the recurrence is bit simple, so iterative solution is expected to make the problem bit harder with time limit.