Solution 4

Group A

1) Problem 3 Consider 7 girls A, B, C, D, E, F, G, starting from A, WLOG, say we have AB and AC:
   - Case I: if we have BC, then D, E, F, G have to form a 4-cycle (e.g. DE, EF, FG, GD). In this case, we have in total $7! \times C_7^3 \times 3 = 529200$.
   - Case II: if we do not have BC, again WLOG, say we have BD, let us consider
     - Subcase (i): if we also have CD, then A, B, C, D form a 4-cycle. This is already covered in Case I.
     - Subcase (ii): if not, again WLOG, say we have CE, then we cannot have DE otherwise there is no arrangement satisfying the requirements. Up to this point, it is easy to see that the only possibility would be A, B, C, D, E, F, G form a 7-cycle.

     In this case, we have in total $7! \times C_6^2 \times 4 \times 3 \times 2 = 1814400$.

   Enventually, we have $529200 + 1814400 = 2343600$ different possibilities.
2) Problem 8

$$\text{success rate} = \frac{\text{\# of successful shoot}}{\text{\# of total shoot}} = \frac{S}{T}$$

   Let $S_0/T_0 < 0.5$ and for some $n \geq 0$, the first time we have $S_{n+1}/T_{n+1} > 0.5$. Consider S_n and T_n, since $S_n/T_n \leq 0.5$, we must have $S_n = S_{n+1} - 1$ and of course $T_n = T_{n+1} - 1$. Having this, we can estimate

$$S_n = S_{n+1} - 1 > \frac{T_{n+1}}{2} - 1 = \frac{T_n}{2} - \frac{1}{2}.$$

   Since both S_n and T_n are integers, we know

$$S_n \geq \left\lceil \frac{T_n}{2} \right\rceil \geq \frac{T_n}{2}.$$

   So, the only possibility is $S_n/T_n = 0.5$.
3) Problem 13

```
#include <iostream>
#include <vector>
using namespace std;

class Solution{
    bool IsValid(vector<vector<char>>& broad, int x, int y){
        // Check columes!
        for (int i=0; i<9; i++) {
            if ((i != x) && (broad[i][y]==broad[x][y])) {
                return false;
            }
        }

        // Check rows!
        for (int j=0; j<9; j++) {
            if ((j != y) && (broad[x][j]==broad[x][y])) {
                return false;
            }
        }

        // Check 3x3 box!
        for (int i=3*(x/3); i<3*(x/3+1); i++) {
            for (int j=3*(y/3); j<3*(y/3+1); j++) {
                if ((i != x) && (j != y) && (broad[i][j]==broad[x][y])) {
                    return false;
                }
            }
        }

        return true;
    }
public:
    Solution(){}
    ~ Solution(){}
    bool SudokuSolver(vector<vector<char>>& broad){
        for (int i=0; i<9; i++) {
            for (int j=0; j<9; j++) {
                // Is the Sudoku still unsolved? Find a vacancy!!!
                if (broad[i][j]=='.') {
                    // Try to fill in 1-9 into vacancy!!!
                    for (int k=0; k<9; k++) {
                        broad[i][j]='1'+k;
                        // DFS!!! since we use SudokuSolver()!!!
                        if (IsValid(broad, i, j && SudokuSolver(broad))) {
                            return true;
                        }
                        broad[i][j]='.';
                    }
                }
                // Cannot solve
                return false;
            }
        }
        // If NO vacancy found=>already solved!!!
        return true;
    }
};
```