```cpp
//
//  There are N dishes, numbered 1,2 ... N. Initially, for each i (1≤i≤N),
//  Dish i has a[i] (1 ⩽ a[i] ⩽ 3) pieces of sushi on it.
//  Taro will perform the following operation repeatedly until all the pieces of sushi are eaten:
//    - Roll a die that shows the numbers 1,2 ... N with equal probabilities, and let i be the outcome.
//      If there are some pieces of sushi on Dish i, eat one of them; if there is none, do nothing.
//  Find the expected number of times the operation is performed before all the pieces of sushi are eaten.
//
//     Time Complexity: O(N^3)
//

#include <bits/stdc++.h>

using namespace std;

int n;  // n is also the "sum of s[i]"
vector<int> mp(4, 0);

const int MAX = 310;
double dp[MAX][MAX][MAX];

double dp_f(int x, int y, int z) {
    if (dp[x][y][z] != -1) {
        return dp[x][y][z];
    }

    if (x == 0 && y == 0 && z == 0) return 0;

    double zero = n - x - y - z;
    double rolls = (n - zero) / n + zero * (2 * n - zero) / (n * (n - zero));
    double current_sum = rolls;

    if (x) {
        double weight = x / (n - zero);
        current_sum += weight * dp_f(x - 1, y, z);
    }

    if (y) {
        double weight = y / (n - zero);
        current_sum += weight * dp_f(x + 1, y - 1, z);
    }

    if (z) {
        double weight = z / (n - zero);
        current_sum += weight * dp_f(x, y + 1, z - 1);
    }

    dp[x][y][z] = current_sum;
    return current_sum;
}

int main() {
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++) {
            for (int k = 0; k < MAX; k++) {
                dp[i][j][k] = -1;
            }
        }
    }

    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        int x;
        scanf("%d", &x);
        mp[x]++;
    }

    cout << setprecision(16) << dp_f(mp[1], mp[2], mp[3]) << endl;
    return 0;
}
```