|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Problem Statement for FixedDiceGameDiv1** | | | | | | |
| http://community.topcoder.com/i/clear.gif | | | | | | |
| http://community.topcoder.com/i/clear.gif | | | | | | |
| |  |  | | --- | --- | | **Problem Statement** | | |  | Alice and Bob are playing a game. Alice rolls **a** identical **b**-sided dice. Bob rolls **c** identical **d**-sided dice. The sides of an n-sided die have numbers 1 through n written on them.  A player's score is the sum of the numbers they rolled on their dice. The player with a strictly higher score wins. It is possible that neither player wins.  You are given the ints **a**, **b**, **c**, and **d**. The players already rolled their dice. If it's not possible for Alice to win, return -1. Otherwise, assume that you don't know what numbers Alice and Bob rolled, but that you know that Alice won the game. Return the expected value of Alice's score (given the above assumption). | |  | | | **Definition** | | |  | |  |  | | --- | --- | | Class: | FixedDiceGameDiv1 | | Method: | getExpectation | | Parameters: | int, int, int, int | | Returns: | double | | Method signature: | double getExpectation(int a, int b, int c, int d) | | (be sure your method is public) | | | |  |  | |  |  | |  | | | **Notes** | | | - | Your return value must have an absolute or relative error smaller than 1e-3. | |  | | | **Constraints** | | | - | **a**, **b**, **c**, **d** will each be between 1 and 50, inclusive. | |  | | | **Examples** | | | 0) |  | |  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | |  | | --- | | 1 | | 2 | | 1 | | 5 | | | Returns: 2.0 | | |  | | --- | | The only way Alice can win is if she rolls a 2. Thus, if we know Alice wins, we know she rolled a 2. | | | | 1) |  | |  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | |  | | --- | | 3 | | 1 | | 1 | | 3 | | | Returns: 3.0 | | |  | | --- | | Alice will always roll a 3. | | | | 2) |  | |  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | |  | | --- | | 1 | | 5 | | 1 | | 1 | | | Returns: 3.4999999999999996 | | |  | | --- | | Alice will not win if she rolls a 1. Thus, if we know she wins, her expected score is (2+3+4+5)/4=7/2. | | | | 3) |  | |  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | |  | | --- | | 2 | | 6 | | 50 | | 30 | | | Returns: -1.0 | | |  | | --- | | No matter what Alice rolls, she will lose. | | | | 4) |  | |  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | |  | | --- | | 50 | | 11 | | 50 | | 50 | | | Returns: 369.8865999182022 | | |  | | --- | |  | | | |  |  |  |  |  |  |

This problem was used for:   
       [Single Round Match 626 Round 1 - Division I, Level One](http://community.topcoder.com/tc?module=ProblemDetail&rd=15859&pm=13239)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Problem Statement for MagicDiamonds** | | | | | | |
| http://community.topcoder.com/i/clear.gif | | | | | | |
| http://community.topcoder.com/i/clear.gif | | | | | | |
| |  |  | | --- | --- | | **Problem Statement** | | |  | You found **n** Magic Diamonds in the mountain. You are now thinking about transfering them to your home. The only way you can transfer Magic Diamonds is to use Transfer Magic one or more times.     The Magic Diamonds are very strange. For any positive integer x you can use Transfer Magic to transfer x Magic Diamonds at once. However, if x is a prime number, the Magic Diamonds will disappear instead of getting transferred. You are not allowed to lose any of the Magic Diamonds, therefore you may never use Transfer Magic on a prime number of Magic Diamonds. Your task is to transfer all Magic Diamonds using Transfer Magic as few times as possible.     You are given a long **n**. Return the minimal number of Transfer Magic usages you need to transfer **n** Magic Diamonds. | |  | | | **Definition** | | |  | |  |  | | --- | --- | | Class: | MagicDiamonds | | Method: | minimalTransfer | | Parameters: | long | | Returns: | long | | Method signature: | long minimalTransfer(long n) | | (be sure your method is public) | | | |  |  | |  |  | |  | | | **Notes** | | | - | A positive integer x is a prime number if and only if it has exactly 2 divisors: 1 and x. Note that 1 is not a prime number. | | - | Your task can always be accomplished. For example, you can use Transfer Magic n times and transfer 1 Magic Diamond each time. | |  | | | **Constraints** | | | - | **n** will be between 1 and 1,000,000,000,000 (10^12), inclusive. | |  | | | **Examples** | | | 0) |  | |  | |  |  | | --- | --- | | |  | | --- | | 2 | | | Returns: 2 | | |  | | --- | | We have to use Transfer Magic twice, each time we transfer 1 Magic Diamond. | | | | 1) |  | |  | |  |  | | --- | --- | | |  | | --- | | 4294967297 | | | Returns: 1 | | |  | | --- | | We just need to use Transfer Magic once, because 4294967297 is not a prime. We have 4294967297 = 641 \* 6700417. | | | | 2) |  | |  | |  |  | | --- | --- | | |  | | --- | | 2147483647 | | | Returns: 2 | | |  | | --- | | This time **n** is a prime, so we have to use Transfer Magic at least twice. We have 2147483647 = 2147400000 + 83647 (83647 = 233 \* 359, which is not a prime), thus the answer is 2. | | | | 3) |  | |  | |  |  | | --- | --- | | |  | | --- | | 1 | | | Returns: 1 | | |  | | --- | |  | | |   This problem statement is the exclusive and proprietary property of TopCoder, Inc. Any unauthorized use or reproduction of this information without the prior written consent of TopCoder, Inc. is strictly prohibited. (c)2010, TopCoder, Inc. All rights reserved. |  |  |  |  |  |  |
| http://community.topcoder.com/i/clear.gif | | | | | | |
| http://community.topcoder.com/i/clear.gif | | | | | | |
| http://community.topcoder.com/i/clear.gif | | | | | | |
| This problem was used for:         [Single Round Match 524 Round 1 - Division I, Level One](http://community.topcoder.com/tc?module=ProblemDetail&rd=14549&pm=11607)         [Single Round Match 524 Round 1 - Division II, Level Two](http://community.topcoder.com/tc?module=ProblemDetail&rd=14549&pm=11607) | | | | | | |
| Problem Statement for HandsShaking | | | | | | |
| http://community.topcoder.com/i/clear.gif | | | | | | |
| http://community.topcoder.com/i/clear.gif | | | | | | |
| |  |  | | --- | --- | | **Problem Statement** | | |  | Consider a meeting of **n** businessmen sitting around a circular table. To start the meeting, they must shake hands. Each businessman shakes the hand of exactly one other businessman. All handshakes happen simultaneously. We say that the shake is perfect if no arms cross each other. Given an int **n**, return the number of perfect shakes that exist for **n** businessmen. See examples for further clarification. | |  | | | **Definition** | | |  | |  |  | | --- | --- | | Class: | HandsShaking | | Method: | countPerfect | | Parameters: | int | | Returns: | long | | Method signature: | long countPerfect(int n) | | (be sure your method is public) | | | |  |  | |  |  | |  | | | **Notes** | | | - | Businessmen are distinguishable. Rotating a perfect shake can yield a different perfect shake (see example 1). | |  | | | **Constraints** | | | - | **n** will be between 2 and 50, inclusive. | | - | **n** will be even. | |  | | | **Examples** | | | 0) |  | |  | |  |  | | --- | --- | | |  | | --- | | 2 | | | Returns: 1 | | |  | | --- | | Two businessmen have only one possibility - just to shake each other's hand. | | | | 1) |  | |  | |  |  | | --- | --- | | |  | | --- | | 4 | | | Returns: 2 | | |  | | --- | | Two out of three possible shakes are perfect.    http://www.topcoder.com/contest/problem/HandsShaking/HandsShaking_4_correct_1.GIF  http://www.topcoder.com/contest/problem/HandsShaking/HandsShaking_4_correct_2.GIF  http://www.topcoder.com/contest/problem/HandsShaking/HandsShaking_4_wrong.GIF | | | | 2) |  | |  | |  |  | | --- | --- | | |  | | --- | | 8 | | | Returns: 14 | | |  | | --- | |  | | |   This problem statement is the exclusive and proprietary property of TopCoder, Inc. Any unauthorized use or reproduction of this information without the prior written consent of TopCoder, Inc. is strictly prohibited. (c)2010, TopCoder, Inc. All rights reserved. |  |  |  |  |  |  |
| http://community.topcoder.com/i/clear.gif | | | | | | |
| http://community.topcoder.com/i/clear.gif | | | | | | |
| http://community.topcoder.com/i/clear.gif | | | | | | |
| This problem was used for:         [Single Round Match 363 Round 1 - Division I, Level One](http://community.topcoder.com/tc?module=ProblemDetail&rd=10777&pm=7868)         [Single Round Match 363 Round 1 - Division II, Level Two](http://community.topcoder.com/tc?module=ProblemDetail&rd=10777&pm=7868) | | | | | | |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Problem Statement for ModeProbability** | | | | | | | | http://community.topcoder.com/i/clear.gif | | | | | | | | http://community.topcoder.com/i/clear.gif | | | | | | | | |  |  | | --- | --- | | **Problem Statement** | | |  | You have a skewed random number generator that outputs the number i with percentage **probs**[i]. Given that you have generated **n** numbers, return the probability (between 0 and 1) that **value** has been generated more times than any of the other numbers. | |  | | | **Definition** | | |  | |  |  | | --- | --- | | Class: | ModeProbability | | Method: | getProb | | Parameters: | int[], int, int | | Returns: | double | | Method signature: | double getProb(int[] probs, int n, int value) | | (be sure your method is public) | | | |  |  | |  |  | |  | | | **Notes** | | | - | The returned value must be accurate to 1e-9 relative or absolute. | |  | | | **Constraints** | | | - | **probs** will contain between 1 and 5 elements inclusive. | | - | Each element of **probs** will be between 1 and 100 inclusive. | | - | The elements of **probs** will sum to 100. | | - | **n** will be between 1 and 15 inclusive. | | - | **value** will be between 0 and N-1 inclusive, where N is the number of elements in **probs**. | |  | | | **Examples** | | | 0) |  | |  | |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | {50,50} | | 2 | | 0 | | | Returns: 0.25 | | |  | | --- | | Two equally occurring numbers. For 0 to occur more than 1 it needs to be generated twice in a row. Hence, the probability is 1/4. | | | | 1) |  | |  | |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | {50,50} | | 9 | | 0 | | | Returns: 0.5 | | |  | | --- | | Since we generate 9 numbers, one number will always occur more times than the other. By symmetry, 0 occurs more frequently with probability 1/2. | | | | 2) |  | |  | |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | {5,50,20,25} | | 15 | | 1 | | | Returns: 0.7947486656372071 | |  | |   This problem statement is the exclusive and proprietary property of TopCoder, Inc. Any unauthorized use or reproduction of this information without the prior written consent of TopCoder, Inc. is strictly prohibited. (c)2010, TopCoder, Inc. All rights reserved. |  |  |  |  |  |  | | http://community.topcoder.com/i/clear.gif | | | | | | | | http://community.topcoder.com/i/clear.gif | | | | | | | | http://community.topcoder.com/i/clear.gif | | | | | | | | This problem was used for:         [2005 TCO Semifinal Room 3 - Division I, Level One](http://community.topcoder.com/tc?module=ProblemDetail&rd=8093&pm=4643) | | | | | | | | | | | | | |
| http://community.topcoder.com/i/clear.gif | | | | | | |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | http://community.topcoder.com/i/clear.gif | | | | | | | | http://community.topcoder.com/i/clear.gif | | | | | | | | **Problem Statement for AutoLoan** | | | | | | | | http://community.topcoder.com/i/clear.gif | | | | | | | | http://community.topcoder.com/i/clear.gif | | | | | | | | |  |  | | --- | --- | | **Problem Statement** | | |  | Auto dealerships frequently advertise tempting loan offers in order to make it easier for people to afford the "car of their dreams". A typical sales tactic is to show you various cars, and then talk in terms of what your monthly payment would be, to say nothing of how much you are actually paying for the car, how much interest you pay, or how long you have to make payments.  A typical auto loan is calculated using a fixed interest rate, and is set up so that you make the same monthly payment for a set period of time in order to fully pay off the balance. The balance of your loan starts out as the sticker price of the car. Each month, the monthly interest is added to your balance, and the amount of your payment is subtracted from your balance. (The payment is subtracted after the interest is added.) The monthly interest rate is 1/12 of the yearly interest rate. Thus, if your annual percentage rate is 12%, then 1% of the remaining balance would be charged as interest each month.  You have been checking out some of the cars at your local dealership, TopAuto. An excited salesman has just approached you, shouting about how you can have the car you are looking at for a payment of only**monthlyPayment** for only **loanTerm** months! You are to return a double indicating the annual percentage rate of the loan, assuming that the initial balance of the loan is **price**. | |  | | | **Definition** | | |  | |  |  | | --- | --- | | Class: | AutoLoan | | Method: | interestRate | | Parameters: | double, double, int | | Returns: | double | | Method signature: | double interestRate(double price, double monthlyPayment, int loanTerm) | | (be sure your method is public) | | | |  |  | |  |  | |  | | | **Notes** | | | - | Because of the way interest is compounded monthly, the actual interest accrued over the course of a year is not necessarily the same as (balance \* yearly interest rate). In fact, it's usually more. | | - | In a real situation, information like this would typically need to be disclosed, but since you aren't at a point of signing any paperwork, the salesman has no legal obligation to tell you anything. | | - | The return value must be within 1e-9 absolute or relative error of the actual result. | |  | | | **Constraints** | | | - | **price** will be between 1 and 1000000, inclusive. | | - | **monthlyPayment** will be between 0 and **price** / 2, inclusive. | | - | **loanTerm** will be between 1 and 600, inclusive. | | - | The resulting interest rate will be between 0 and 100, inclusive. | |  | | | **Examples** | | | 0) |  | |  | |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | 6800 | | 100 | | 68 | | | Returns: 1.3322616182218813E-13 | | |  | | --- | | Noting that 68 payments of 100 equals the total price of 6800, so there is no interest. | | | | 1) |  | |  | |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | 2000 | | 510 | | 4 | | | Returns: 9.56205462458368 | | |  | | --- | | Here, we do pay a little interest. At 9.562% annual interest, that means each month we pay 0.7968% of the balance in interest. Our payment schedule looks like this:  Month | + Interest | - Payment | = Balance  ------------------------------------------  | | | 2000.00  1 | 15.94 | 510.00 | 1505.94  2 | 12.00 | 510.00 | 1007.94  3 | 8.03 | 510.00 | 505.97  4 | 4.03 | 510.00 | 0.00 | | | | 2) |  | |  | |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | 15000 | | 364 | | 48 | | | Returns: 7.687856394581649 | | |  | | --- | | This is similar to what purchasing a new car with no money down might look like, if you make payments for 4 years. | | |   This problem statement is the exclusive and proprietary property of TopCoder, Inc. Any unauthorized use or reproduction of this information without the prior written consent of TopCoder, Inc. is strictly prohibited. (c)2010, TopCoder, Inc. All rights reserved. |  |  |  |  |  |  | | http://community.topcoder.com/i/clear.gif | | | | | | | | http://community.topcoder.com/i/clear.gif | | | | | | | | http://community.topcoder.com/i/clear.gif | | | | | | | | This problem was used for:         [Single Round Match 258 Round 1 - Division I, Level One](http://community.topcoder.com/tc?module=ProblemDetail&rd=7993&pm=3970)         [Single Round Match 258 Round 1 - Division II, Level Two](http://community.topcoder.com/tc?module=ProblemDetail&rd=7993&pm=3970) | | | | | | | | | | | | | |
| http://community.topcoder.com/i/clear.gif | | | | | | |