Gift Shop

Store all souvenirs' names and total cost ($Q \cdot P$) in arrays and determine the min and max total cost. Loop through the total cost array, and on one line print out the names of the souvenirs whose total cost is equal to the max total cost. Do the same thing for the min total cost on the next line. Make sure to separate the names by a single space (trailing whitespace does not matter on HackerRank).

Hatching

The number of days can be solved with a simple formula:

$$D = \lceil (T - S) \cdot Q \div P \rceil$$

Make sure to use the ceiling function because the problem states to round up to the nearest integer.

Pokémon Party

Because there are no ties in any of the mini-games, there are $N!$ unique permutations of how Poké Snacks can be distributed with N Pokémon playing. With N mini-games, this brings us to $(N!)^N$ possible scenarios, and because N is at most 4, that means there are at most 331,776 scenarios needed to be checked. This can easily be done in the given time limit using recursion.

First store all the unique permutations of $\{1, \ldots, N\}$ to avoid recomputing it for every mini-game. With recursion, keep track of the current state of how many Poké Snacks each Pokémon has, S, and the amount of mini-games leftover, L. Initially all Pokémon will start with 0 Poké Snacks and will play N mini-games. For each mini-game, loop through all the unique permutations stored and distribute it to S. Recurse on each new state and decrease L by 1 to indicate a mini-game has been completed. Once $L = 0$, determine the max Poké Snacks any Pokémon has and check if Pikachu has that amount (determining who is Pikachu is arbitrary but must be consistent throughout all scenarios). If Pikachu has the most or the same amount as the most, then he is considered a winner. Count the number of times this occurs.

Angry Bird

With some basic knowledge in energy conservation and kinematics, this problem becomes as simple as launching birds across your screen! To solve this problem, we'll use the following relations:

$$KE = \frac{1}{2}mv^2 \tag{1}$$

$$PE = \frac{1}{2}kx^2 \tag{2}$$

$$\Delta y = V_0 + \frac{1}{2}at^2 \tag{3}$$

$$v_f = v_0 + at \tag{4}$$

$$v_x = v\cos\theta, v_y = v\sin\theta \tag{5}$$

$$x = v_x t \tag{6}$$

Let's split this problem up into multiple parts. First, we'll calculate the velocity at which Pidove leaves the slingshot. Using equations (1) and (2), we easily rearrange for $v = kx^2/m$. In order to determine distance using equation (6), however, we must first find the time it takes for Pidove to hit the ground. To do this, we can split Pidove's trajectory into two parts: the first being the path to the apex, and the second being the path down to the ground. We calculate time for the first path using equations (4) and (5). We find the height of the apex using equation (3) and adding the initial starting height of h. Using this, we can find the time it takes for Pidove to hit the ground using equation (4), assuming that v_0 is 0. Using the times for the separate paths, we can then calculate the final distance using equation (6).

Note: use `Math.toRadians()` or an equivalent precaution whenever you're dealing with angles in degrees!

Base Scoring

Keep track of the largest sum and the current winner as while looping through all coordinators. For each coordinator, convert all three judges' scores to octal and sum the digits. Converting to octal can be done via many ways such as the use of modulo 8 or `Integer.toOctalString()`. If a coordinator's sum is greater than the current largest sum, update the largest sum and current winner. After checking all coordinators, print out the winner.

Call For Help!

Using an if statement, check if $P > L$, $P < L$, or $P = L$ and print according to the problem.

My Poké Academia

Because the average score of the students is strictly less than P, it is not possible to obtain P when $P = 100$. Therefore, the only scenario that would be `impossible` would be when $P = 100$. To solve the rest of the problem, obtain the initial sum of Uked's scores, T, and keep track of the current amount of scores he has, A, which is initially equal to N. In order to determine the least amount of results to raise his average score to P, keep adding scores of 100 to T while also increasing A by 1 until $T \div A \geq P$. Afterwards, the answer is just $A - N$.

Alternatively, knowing that Uked would need to keep adding scores of 100 to his results, a formula could've been derived:

$$\frac{T + 100x}{N + x} = P$$
$$x = \left\lceil \frac{NP - T}{100 - P} \right\rceil$$

Power Plant Scheming

Using floodfill can easily determine the maximum area in the grid. However, standard floodfill using recursion will not work given the constraints of the problem as it could produce a StackOverflowError. To solve this problem without recursion, we can utilize a Stack or Queue and create a custom class (or even an array) to store the current row and column we're located at in the map. For each location, travel either up, down, left, or right if and only if that location hasn't been visited yet, and it is an empty area (`'.'`). Perform the floodfill everywhere and each time count the number locations visited. Determine the max locations visited.

Daycare Drop-off

Create an array, arr, where $arr[d]$ represents the amount of Pokémon at the daycare on a given day d. For each A_i and B_i, increase arr from A_i to $B_i - 1$ by 1 to indicate a Pokémon is at the daycare during that interval. When processing queries, read each day, D, and the answer is simply $\lceil arr[D] \div K \rceil$. This, however, only solves a few of the test cases within the time limit.

To solve all test cases within the time limit, we can use a prefix sum on the array, arr. For each A_i and B_i, we increase $arr[A_i]$ by 1 and decrease $arr[B_i]$ by 1. Afterwards, perform the prefix sum on the array by looping from 1 to the end of the array, where each value at index i is equal to itself plus the previous value at index $i - 1$. Processing queries should remain the same where each day, D, should result in the answer $\lceil arr[D] \div K \rceil$.

Do You Like Chespin and His Poké Puffs?

Simply keep track of the largest length and width that can be made from the cuts. This can be done by looping over each of the horizontal and vertical cuts once. For the horizontal, keep track of the previous horizontal cut, p, and obtain the maximum width of $h_i - p$. Similarly for the vertical cuts, keep track of the previous vertical cut, p, and find the maximum length of $v_i - p$. Be sure to check edge cases such as the maximum widths being h_1 or $R - h_H$ and the maximum lengths being v_1 or $C - v_V$. Afterwards, multiply the maximum width and maximum length to get the answer.

Elimination

To solve this, it requires sorting the 6 different stats separately. Creating a custom Comparable class or some sort of pairing to pair a Pokémon's name with its stat will be needed. Sort the stats in ascending order, and remove the first K Pokémon in each category. A TreeSet of Strings can be used to store the unique names of the eliminated Pokémon in alphabetical order. Iterate over the TreeSet and print out the names. Remember to use `long` because of stats being as large as 10^{10}.

Naku

The simplest solution involves using Regex and String's `replaceAll()` method. Alternatively, for each cry, create a new String, S, initialized with the first character of the corresponding cry. Loop from 1 to the end of the cry and check whether or not to add a character to S. If the current character is not a vowel add it to S, or if the current character is a vowel and not the same as the previous character in S, then add it to S. Afterwards, print S.