Mastermind: The ultimate foe (mastermind)

The game, called “Matermind” is played by two players: code-creator and codebreaker. The code-creator creates a code, consisting of n numbers in the interval from 0 to m-1 (inclusive). Repeating numbers are allowed. The codebreaker tries to break that code. He makes a guess, consisting of n numbers, every in the interval from 0 to m-1. After that he wants a hint from the code-creator. The code-creator answers him with two numbers: c and p. The numbers show that codebreaker guessed c numbers in their correct positions, and another p numbers that are in the code but are in the wrong position. The goal of the game is to break the code (guess it) with as less moves as possible. Here is an example (n=4,m=6):

1. Code-creator chose the code (3, 3, 5, 4).
2. Codebreaker guesses (3, 3, 3, 3)
3. The answer of the code-creator is c=2,p=0. The two threes in the beginning of the code are in their correct positions. Note that the other two trees are not “existing but in wrong positions” because there are only two threes in the code.
4. Codebreaker guesses (4, 4, 3, 3)
5. The answer is c=0,p=3. This time one of the fours and the two threes exist in the code, but are in the wrong positions.
6. The next guess is (3, 3, 4, 5)
7. The answer is c=2,p=2
8. Next guess is (3, 3, 5, 4)
9. The answer is c=4,p=0
10. The game ends here after 4 guesses from the codebreaker.

That is one round of the game. For the next round the code-creator creates a new code, that the codebreaker has to break again. One game (with fixed n and m) is played in a few rounds.

It is important to note, that the code-creator is not obligated to be completely fair in his game, meaning the code doesn't have to be created in the beginning. The code-creator can change the code in the course of the round, to make the game harder. His answers shouldn't contradict with the code. In other words, coming back in the beginning of the game, all answers must match the same code.

Task

Your task is to create that *ultimate foe.* The grading system will be the codebreaker, while your program will play the role of the code-creator. The system will call your procedure Init with two parameters – n and m. It is guaranteed, that this function will be called only once in the beginning of the execution of your program.

After that your procedure NewRound will be called several times without any parameters. That starts a new round and can be done a lot of times, but only after the call of Init and after the grading system has guessed your code in the last round.

After the beginning of a new round, your program (the code-creator) must give answers to the grading system (the codebreaker). That will be done by the grading system, that will call your procedure Hint, passing her three parameters. First of them is an array of n integer numbers, in the interval from 0 to m-1. This array is the guess of the codebreaker. The second and third parameters are c and p, that are passed by place (Pascal) or as pointers (C/C++). Your program should return the amount of numbers that are in their correct positions as c and the amount of numbers that are in wrong positions as p, the way it is described above. The answers shouldn't contradict!

Every round ends, when Hint returns value of n in its parameter c and value of 0 in its parameter p, which means that the codebreaker broke the code. The game ends, when the codebreaker stops calling NewRound and terminates your program.

Grading

In one execution of your program (in one game) the amount of guesses made by the grading system to break your code will be saved. The smallest of those

numbers will be used to determine the score you will get on the corresponding test. Let’s denote that number with t. There will be only one test in each subtask, so that will be the points for the whole subtask. For every subtask a number k is determined. If t=1, you will get 0 points. If t=2, you will get 7% of the points for the subtask. If t>=k, you will get full points for the subtask. Every numbers between t=2 and t=k will be scored linearly between 7% and 100% of the points for the subtask.

Subtasks

Subtask 1 (12 points): n=2, m=4, k=4

Subtask 2 (21 points): n=3, m=10, k=7

Subtask 3 (23 points): n=4, m=6, k=5

Subtask 4 (21 points): n=5, m=5, k=5

Subtask 5 (23 points): n=6, m=3, k=5