Rygar (E/S Python)

Rygar, o guerreiro, está diante de uma horda de monstros. Embora Rygar seja muito forte, ele não sabe se poderá vencer todos. Mesmo assim, Rygar irá lutar o máximo que puder.

No começo do combate, Rygar possui 100 pontos de Energia. Já os monstros são variados e alguns são mais resitentes que outros, portanto a Energia inicial de cada monstro deve ser lida do caso de teste.

A Força de Rygar é 40. Isso significa que, quando Rygar ataca um monstro, o monstro perde 40 pontos de Energia. A Força de cada monstro pode ser diferente e deve ser lida do caso de teste.

O combate ocorre em turnos. No início de cada turno, Rygar escolhe um dos monstros para combater. Rygar ataca primeiro e, em seguida, é o monstro quem ataca. Enquanto eles estão lutando, os outros monstros apenas observam. O embate dura até um dos dois perder toda a Energia.

Se Rygar vencer e ainda houver monstro para lutar, então escolherá outro monstro e um novo combate em turnos terá início. A Energia de Rygar não é reposta entre as batalhas. Rygar continuará escolhendo monstros até derrotar todos ou ser derrotado.

Considere que Rygar sempre faz a melhor escolha, de modo que conseguirá derrotar o maior número possível de monstro antes de sucumbir aos ferimentos. Escreva um programa para determinar se Rygar sobreviverá ao combate ou, se a vitória for impossível, qual é o maior número de monstros que ele conseguirá derrotar antes de sucumbir aos ferimentos.

Como entrada, seu programa deve ler uma lista de tuplas $((e_1,f_1),(e_2,f_2),...,(e_N,f_N))$((e1,f1),(e2,f2),…,(eN,fN)) que descrevem os monstros. Cada par (e_i,f_i)(ei,fi) é uma tupla em Python que indica, respectivamente, a Energia e a Força de cada monstro.
Como saída, imprima o número de monstros que Rygar consegue derrotar e a quantidade de Energia de Rygar ao final do combate. Não se esqueça que Rygar sucumbe se sua Energia chegar a zero. O valor da Energia nunca pode ser negativo.

Dicas

Este é um desafio que envolve ordenação. Você consegue enxergar a “pegadinha”?

Para resolver o exercício, pode ser necessário arrendodar valores para cima ou para baixo, dependendo da forma como você fizer os cálculos. As funções abaixo podem ser usadas para isso:

- A função **ceil(x)** retorna o valor de **x** arredondado para cima (teto). Por exemplo **ceil(1.1)** é **2.0**. Para usar esta função, acrescente a linha `from math import ceil` no começo do seu código.
- A função **round(x)** retorna o valor de **x** arredondado para o inteiro mais próximo. Por exemplo **round(1.5)** é **2.0**, mas **round(1.4)** é **1.0**
- A função **floor(x)** retorna o valor de **x** arredondado para baixo (chão). Por exemplo, **floor(1.9)** é **1.0**. Para usar esta função, acrescente a linha `from math import floor` no começo do seu código.

Exemplos de Entrada e Saída

Entrada	`((60, 30), (40, 80), (60, 40), (70, 20))`
Saída	`4 10`
Entrada	`((60, 60), (20, 180), (60, 40), (70, 20))`
Saída	`3 0`
Entrada	`((300, 10), (500, 2), (500, 20), (50, 200))`
Saída	`2 0`