Você conhece o *jogo do NIM*? Nesse jogo, **n** peças são inicialmente dispostas numa mesa ou tabuleiro. Dois jogadores jogam alternadamente, retirando pelo menos 1 e no máximo **m** peças cada um. Quem tirar as últimas peças possíveis ganha o jogo.

Existe uma estratégia para ganhar o jogo que é muito simples: ela consiste em deixar sempre múltiplos de **(m+1)** peças ao jogador oponente.

**Objetivo**

Você deverá escrever um programa na linguagem Python, versão 3, que permita a uma "vítima" jogar o NIM contra o computador. O computador, é claro, deverá seguir a estratégia vencedora descrita acima.

Sejam **n** o número de peças inicial e **m** o número máximo de peças que é possível retirar em uma rodada. Para garantir que o computador ganhe sempre, é preciso considerar os dois cenários possíveis para o início do jogo:

* Se **n** é múltiplo de **(m+1)**, o computador deve ser "generoso" e convidar o jogador a iniciar a partida com a frase "Você começa!"
* Caso contrário, o computador toma a iniciativa de começar o jogo, declarando "Computador começa!"

Uma vez iniciado o jogo, a estratégia do computador para ganhar consiste em deixar sempre um número de peças que seja múltiplo de **(m+1)** ao jogador. Caso isso não seja possível, deverá tirar o número máximo de peças possíveis.

Seu trabalho, então, será implementar o Jogo e fazer com que o computador se utilize da estratégia vencedora.

**Seu Programa**

Com o objetivo do EP já definido, uma dúvida que deve surgir nesse momento é como modelar o jogo de forma que possa ser implementado em Python 3 correspondendo rigorosamente às especificações descritas até agora.

Para facilitar seu trabalho e permitir a correção automática do exercício, apresentamos a seguir um modelo, isto é, uma descrição em linhas gerais de um conjunto de funções que resolve o problema proposto neste EP. Embora sejam possíveis outras abordagens, é preciso atender exatamente o que está definido abaixo para que a correção automática do trabalho funcione corretamente.

O programa deve implementar:

* Uma função **computador\_escolhe\_jogada** que recebe, como parâmetros, os números **n** e **m** descritos acima e devolve um inteiro correspondente à próxima jogada do computador (ou seja, quantas peças o computador deve retirar do tabuleiro) de acordo com a estratégia vencedora.
* Uma função **usuario\_escolhe\_jogada** que recebe os mesmos parâmetros, solicita que o jogador informe sua jogada e verifica se o valor informado é válido. Se o valor informado for válido, a função deve devolvê-lo; caso contrário, deve solicitar novamente ao usuário que informe uma jogada válida.
* Uma função **partida** que não recebe nenhum parâmetro, solicita ao usuário que informe os valores de **n** e **m** e inicia o jogo, alternando entre jogadas do computador e do usuário (ou seja, chamadas às duas funções anteriores). A escolha da jogada inicial deve ser feita em função da estratégia vencedora, como dito anteriormente. A cada jogada, deve ser impresso na tela o estado atual do jogo, ou seja, quantas peças foram removidas na última jogada e quantas restam na mesa. Quando a última peça é removida, essa função imprime na tela a mensagem "O computador ganhou!" ou "Você ganhou!" conforme o caso.

Observe que, para isso funcionar, seu programa deve sempre "lembrar" qual é o número de peças atualmente no tabuleiro e qual é o máximo de peças a retirar em cada jogada.

**Cuidado:** o corretor automático não funciona bem se você tiver alguma chamada a input() antes da definição de todas as funções do jogo (a menos que essa chamada esteja dentro de uma função). Se seu programa usar input() sem que ele esteja dentro de alguma função, coloque-o no final do programa.

**Campeonatos**

Como todos sabemos, uma única rodada de um jogo não é suficiente para definir quem é o melhor jogador. Assim, uma vez que a função **partida** esteja funcionando, você deverá criar uma outra função chamada **campeonato**. Essa nova função deve realizar três partidas seguidas do jogo e, ao final, mostrar o placar dessas três partidas e indicar o vencedor do campeonato. O placar deve ser impresso na forma

**Placar: Você ??? X ??? Computador**

**Execução**

Dado que é possível jogar partidas individuais ou campeonatos, seu programa deve começar solicitando ao usuário que escolha se prefere jogar apenas uma partida (opção **1**) ou um campeonato (opção **2**).

**Atenção:** o corretor automático vai verificar se você está utilizando exatamente as mensagens pedidas, como "Você começa!", "O computador ganhou!" etc. Deixe para usar a sua criatividade em outros lugares!

Veja um exemplo de como deve funcionar o jogo:

$ > python3 jogo\_nim.py

Bem-vindo ao jogo do NIM! Escolha:

1 - para jogar uma partida isolada

2 - para jogar um campeonato 2

Voce escolheu um campeonato!

\*\*\*\* Rodada 1 \*\*\*\*

Quantas peças? 3

Limite de peças por jogada? 1

Computador começa!

O computador tirou uma peça.

Agora restam 2 peças no tabuleiro.

Quantas peças você vai tirar? 2

Oops! Jogada inválida! Tente de novo.

Quantas peças você vai tirar? 1

Você tirou uma peça.

Agora resta apenas uma peça no tabuleiro.

O computador tirou uma peça.

Fim do jogo! O computador ganhou!

\*\*\*\* Rodada 2 \*\*\*\*

Quantas peças? 3

Limite de peças por jogada? 2

Voce começa!

Quantas peças você vai tirar? 2

Voce tirou 2 peças.

Agora resta apenas uma peça no tabuleiro.

O computador tirou uma peça.

Fim do jogo! O computador ganhou!

\*\*\*\* Rodada 3 \*\*\*\*

Quantas peças? 4

Limite de peças por jogada? 3

Voce começa!

Quantas peças você vai tirar? 2

Voce tirou 2 peças.

Agora restam 2 peças no tabuleiro.

O computador tirou 2 peças.

Fim do jogo! O computador ganhou!

\*\*\*\* Final do campeonato! \*\*\*\*

Placar: Você 0 X 3 Computador

**Dicas sobre o Jogo do NIM**

O próximo vídeo, **Dicas sobre o Jogo do NIM**, explica detalhes importantes do jogo, como erros comuns e exemplos de mensagem do corretor automático, e pode ser de grande ajuda no desenvolvimento do programa.