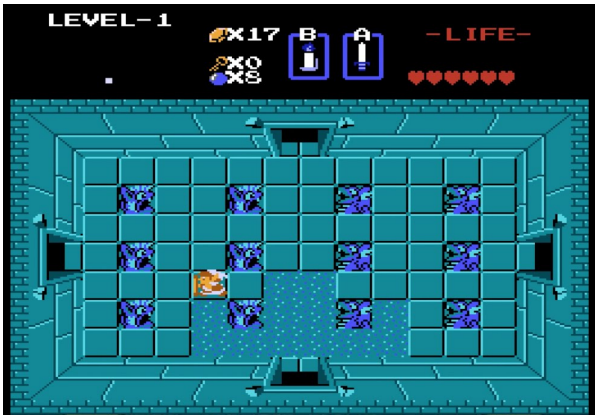
MC102 - Algoritmos e Programação de Computadores

Lab 11

Data da Primeira Chance: 12 de junho de 2023
Peso: 3

Havia uma terra mágica e perigosa conhecida como Hyrule, repleta de masmorras com monstros ferozes e tesouros escondidos. Nessa terra, habitava um herói chamado Link, conhecido por sua coragem e habilidade em enfrentar desafios. Um dia, o reino de Hyrule foi ameaçado por uma terrível força das trevas que estava se espalhando pela masmorra da cidade. O rei de Hyrule convocou Link, confiando-lhe a missão de explorá-la, derrotar os monstros e recuperar os objetos mágicos que poderiam ajudar a salvar o reino.



Fonte: https://fandomspot.com/loz-dungeon-music

As masmorras são salas retangulares compostas por linhas e colunas. Cada posição pode conter diversos objetos ou monstros, assim como o próprio Link. Um exemplo pode ser visto na imagem acima à direita.

Link iniciou sua jornada com sua espada afiada, seu escudo resistente e um mapa da masmorra. Para acessar a masmorra, Link teve que enfrentar uma maldição que restringia seus movimentos e o impedia de se mover livremente por ela. A maldição exigia que, após entrar na masmorra, Link se mova até a última linha, partindo de sua posição inicial. Em seguida, a maldição só permite movimentação seguindo as restrições:

- Mover da esquerda para direita em linhas de número ímpar;
- Mover da direita para esquerda em linhas de número par.

Ao chegar no limite lateral da masmorra, Link deve ir à linha imediatamente acima da linha atual. É importante ressaltar que Link não pode se movimentar pelas diagonais, e que a primeira linha é considerada com índice zero. Assuma que é sempre possível chegar à saída da masmorra.

Link encontra objetos valiosos à medida que Link avança, sempre coletando todos pela frente. Alguns objetos concedem mais vida, fortalecendo sua resistência para enfrentar batalhas difíceis. Outros objetos aumentam seu poder de ataque, permitindo que derrotem monstros mais facilmente. Cada objeto pode ser um dos dois tipos disponíveis:

- v (vida): adiciona um certo valor à vida atual de Link. Este valor pode ser tanto positivo ou negativo, sendo possível que ele morra após coletar algum item.
- d (dano): adiciona um certo valor ao dano atual de Link. Pode ser positivo ou negativo. Em caso de valores menores que 1 (um), o dano mínimo permitido deve ser fixado em 1.

Link também teve que enfrentar monstros poderosos, cada qual possui um determinado valor de vida e dano de ataque. Assim como os objetos, cada monstro possui um tipo específico que determinava o seu padrão de movimentação. Os tipos disponíveis são:

- U: move para uma posição imediatamente acima da atual
- D: move para uma posição imediatamente abaixo da atual
- L: move para uma posição à esquerda da atual
- R: move para uma posição à direita da atual

Caso a próxima posição do monstro, conforme o seu tipo de movimentação, seja para uma posição fora da matriz, o mesmo deve ficar parado.

Link entra em combate com um monstro quando ambos se encontram na masmorra (ocupam a mesma posição). No combate, ele sempre ataca primeiro e recebe o dano em seguida (caso o monstro sobreviva). O turno de combate acontece uma vez naquela posição específica, ou seja, caso os dois envolvidos sobrevivam ao ataque, ambos seguem sua movimentação normalmente na próxima rodada. Caso Link ocupe a mesma posição que dois ou mais monstros, a luta acontece contra todos os inimigos, um de cada vez. Se a próxima movimentação de Link coincidir com a próxima posição do monstro, ambos se enfrentam novamente. Por exemplo, quando Link e 2 monstros estiverem na mesma posição, a ordem de combate será:

- Link atacará o monstro 1. Se o monstro sobreviver, este atacará Link;
- Se Link sobreviver ao combate anterior, ele atacará o monstro 2. Caso o monstro sobreviva, este atacará Link.

Adote a ordem de combate entre os monstros como a mesma da entrada.

Em casos que a posição atual coincide com a posição de um ou mais monstros e um ou mais objetos, adote que Link primeiro coleta cada objeto e depois muda para o ato de combate. Por

fim, o objetivo de Link é atingir uma posição específica no mapa que vai levá-lo até a saída masmorra, livrando assim a cidade de todo o perigo que lá havia por um tempo.

Entrada

A entrada do programa é constituída pelas seguintes informações, em ordem:

- Dois inteiros Vp e Dp, indicando a vida e o dano inicial do Link, respectivamente;
- Dois inteiros N e M, indicando o número de linhas e colunas do mapa, respectivamente;
- Dois inteiros I_x, I_y indicando a posição inicial do personagem na masmorra;
- Em seguida, recebe-se dois inteiros F_x, F_y, indicando a posição (F_x, F_y) de saída da masmorra;
- Um inteiro Q_m indicando o número de monstros no mapa;
- Q_m linhas contendo 5 informações sobre cada monstro: vida, ataque, tipo e posição (P_x, P_y) inicial do monstro;
- Um inteiro B indicando o número de objetos no mapa;
- B linhas contendo 5 informações sobre cada objeto: nome, tipo, posição (O_x, O_y) e status do objeto. O status pode ser entendido como o valor a ser somado à vida ou ao ataque de Link, dependendo do tipo do objeto. Este valor pode ser positivo ou negativo.

Saída

Deve-se imprimir o estado do mapa ao final da movimentação do personagem e dos monstros e de qualquer resolução de combate, assim como coleta de itens. ***Assuma que Link anda sempre primeiro que os monstros.*** A posição atual do personagem deve ser indicada pela letra P, posição de chegada pelo símbolo *, os monstros pelo seu respectivo tipo (U, D, R, L) , assim como os objetos (v ou d). Se o personagem morre após um certo combate ou após a obtenção de algum objeto, deve-se imprimir X no lugar de P e automaticamente a jornada deve parar. Os espaços das masmorras sem nenhum personagem, objeto, monstro, ou que seja posição de chegada deve ser mostrado como '.' (ponto final).

Se a posição atual do Link coincidir com a de algum objeto, deve-se imprimir:

- *[tipo_objeto]Personagem adquiriu o objeto nome_objeto com status de valor,* onde ***tipo_objeto*** refere-se ao tipo do objeto (v ou d); ***nome_objeto*** refere-se ao nome do objeto encontrado e ***valor*** refere-se ao valor de referência daquele objeto. Um exemplo de saída: [***v]Personagem adquiriu o objeto joia com status de 10***.

Ao entrar em combate contra algum monstro, deve-se imprimir:

- *O Personagem deu total_dano de dano ao monstro na posicao posicao_monstro*, onde ***total_dano*** é igual ao dano dado ao monstro. Se o dano for maior que vida atual, mostre o valor que deixa a vida do monstro igual a zero. Já ***posicao_monstro*** refere-se à posição em que Link encontrou o monstro para combate. Caso o monstro sobreviva, devemos imprimir o resultado do combate do monstro com o personagem: *O Monstro*

deu X de dano ao Personagem. Vida restante = restante, onde ***X*** representa o dano do monstro enfrentado e ***restante*** o valor da vida restante do personagem. Se o dano causado pelo monstro for maior do que a vida de Link, mostre que o dano foi o valor necessário para deixar a vida de Link zerada.

Ao imprimir o estado do mapa, consideramos a seguinte ordem de prioridades:

- Se dois ou mais monstros estiverem na mesma posição em que Link, imprime-se a letra P se Link conseguir resistir a todos os combates, ou X caso Link acabe morrendo durante a jornada;
- Se Link estiver na posição de chegada, deve-se imprimir P no lugar de * (símbolo da chegada);
- O símbolo * terá prioridade sobre qualquer outro tipo, exceto o de Link. Sendo assim, quando algum monstro estiver na posição de chegada, deve-se imprimir o símbolo da chegada e não o do monstro;
- Quando monstros e objetos estiverem na mesma posição, considere que monstros possuem prioridade neste caso e deve-se imprimir o tipo do último monstro naquela posição considerando a ordem da entrada;
- Se dois ou mais monstros de diferentes tipos ocuparem a mesma posição, imprima o tipo do último monstro seguindo a ordem de entrada.

Caso Link chegue até a posição desejada, imprima a seguinte frase:

- Chegou ao fim!

Assuma que a posição inicial de Link nunca será a mesma de nenhum monstro ou objeto.

Exemplos

Exemplo 1:

Entrada

```
5 10
4 4
0,1
0,3
3
100 5 U 1,0
110 5 R 2,0
100 5 R 3,0
2
joia d 1,2 -1
chocolate v 2,2 5
```

Saída

```
. P . *
U . d .
R . v .
R . . .

U . . *
. P d .
. R v .
. R . .

U . . *
. . d .
. P R .
. . R .

U . . *
. . d .
. . v R
. P . R

U . . *
. . d .
. . v R
. . P R

O Personagem deu 10 de dano ao monstro na posicao (3, 3)
O Monstro deu 5 de dano ao Personagem. Vida restante = 0
U . . *
. . d .
. . v R
. . . X
```

Exemplo 2:

Entrada

```
100 50
3 3
0,1
0,0
2
100 5 D 0,0
110 5 D 0,0
2
joia d 0,2 2
chocolate v 0,2 5
```

Saída

```
* P v
. . .
. . .

* . v
D P .
. . .

* . v
. . .
D P .

O Personagem deu 50 de dano ao monstro na posicao (2, 0)
O Monstro deu 5 de dano ao Personagem. Vida restante = 95
O Personagem deu 50 de dano ao monstro na posicao (2, 0)
O Monstro deu 5 de dano ao Personagem. Vida restante = 90
* . v
. . .
P . .

* . v
P . .
D . .

* . v
. P .
D . .

* . v
. . P
D . .

[d]Personagem adquiriu o objeto joia com status de 2
[v]Personagem adquiriu o objeto chocolate com status de 5
* . P
. . .
D . .

* P .
. . .
D . .

P . .
. . .
D . .

Chegou ao fim!
```

Exemplo 3:

Entrada

```
500 10
3 5
0,1
0,4
2
10 2 U 1,0
11 2 L 2,0
2
joia d 1,2 20
chocolate v 2,2 50
```

Saída

```
. P . . *
U . d . .
L . v . .

U . . . *
. P d . .
L . v . .

U . . . *
. . d . .
L P v . .

O Personagem deu 10 de dano ao monstro na posicao (2, 0)
O Monstro deu 2 de dano ao Personagem. Vida restante = 498
U . . . *
. . d . .
P . v . .

U . . . *
P . d . .
L . v . .

U . . . *
. P d . .
L . v . .

[d]Personagem adquiriu o objeto joia com status de 20
U . . . *
. . P . .
L . v . .

U . . . *
. . . P .
L . v . .
```

```
U . . . *
. . . . P
L . v . .

U . . . P
. . . . .
L . v . .

Chegou ao fim!
```

Regras e Avaliação

Nesse laboratório, você não pode usar bibliotecas (isto é, o comando *import*). Exceto pelas bibliotecas:

- **typing** para melhorar a clareza e escrita do código**;**
- **dataclass** para definir o conceito de classes no código**.**

Você deve usar classes novas (definidas por você) para representar os elementos do jogo, algo que será importante para avaliar a qualidade do seu código.

Seu código será avaliado não apenas pelos testes do CodePost, mas também pela qualidade. Dentre os critérios subjetivos de qualidade de código iremos analisar nesse laboratório: o uso apropriado de **métodos** e **classes**, e de documentação; a escolha de bons nomes de funções e variáveis; a ausência de diversos trechos de código repetidos desnecessariamente. Note, porém, que essa não é uma lista exaustiva, pois outros critérios podem ser analisados dependendo do código apresentado visando mostrar ao aluno como o código poderia ser melhor.

Os casos de teste estão disponíveis no link.

Submissão

Você deverá submeter no CodePost, na tarefa Lab 11, um arquivo com o nome `lab11.py`. Você pode enviar arquivos adicionais caso deseje para serem incluídos por `lab11.py`. Após a correção da primeira entrega, será aberta uma tarefa Lab 11 - Segunda Chance, com prazo de entrega apropriado.