

Departamento de Informática

Introdução à Programação Torneio da Glória

Enunciado do Segundo Trabalho



Ano letivo 2022/23

Preâmbulo

O trabalho é realizado em grupo de **dois alunos do mesmo turno prático**. Só serão aceites trabalhos de grupos de alunos de turnos práticos diferentes mediante autorização prévia, sendo necessário apresentar uma boa justificação para a autorização ser concedida.

O trabalho é entregue no Mooshak, que aceita submissões até às **20h00** do dia **10 de dezembro de 2022**. Leia este enunciado com a máxima atenção, para perceber muito bem o problema e todos os detalhes sobre o concurso do Mooshak e os critérios de avaliação do trabalho.

1 Conceitos e Objetivo do Trabalho

Depois de tantos anos a jogar o Jogo da Glória, vão ser organizados Torneios da Glória. Num *torneio* em que participam J jogadores, realizam-se $J - 1$ jogos, em sequência. Todos os jogadores participam no primeiro jogo mas, como é eliminado um jogador no decorso ou no fim de cada jogo, o número de jogadores *em jogo* vai diminuindo. O vencedor do último jogo ganha o torneio.

Todos os jogos de um torneio usam o mesmo *tabuleiro*. O tabuleiro tradicional (apresentado na Figura 1), é composto por 90 casas, numeradas de 1 a 90. Cada jogador tem uma *marca* (como as ilustradas na Figura 2), que (em cada jogo) começa na casa 1. Quando é a sua vez de jogar, o jogador lança dois dados para se determinar quantas casas a marca avança. Ganha o jogo quem conseguir atingir primeiro a casa 90 ou quem, ao lançar os dados pela primeira vez no jogo, obtiver 6 pintas num dado e 3 pintas no outro.

Para haver mais emoção, há casas especiais, chamadas *multas*, *precipícios* e *pássaros*. Se a marca para numa multa, o jogador fica impedido de lançar os dados durante algumas jogadas. A penalização sofrida quando a marca para num precipício varia muito com o precipício. Por exemplo, o *caranguejo* faz a marca recuar (em vez de avançar), o *inferno* obriga a marca a regressar à casa 1 e a *morte* expulsa o jogador (impedindo-o de continuar a jogar). Ao contrário das multas e dos precipícios, parar numa casa com pássaro é uma sorte, porque se avança imediatamente mais 9 casas.

A *classificação* dos jogadores é determinada pelo momento em que eles são eliminados (o primeiro jogador eliminado é o último classificado). Enquanto os jogadores estão em jogo, o seu desempenho depende, essencialmente, do número de jogos ganhos.

O objetivo deste trabalho é programar em Java uma versão muito simplificada do Torneio da Glória. Com a informação sobre o número de jogadores e as cores das suas marcas, o tabuleiro e a sequência de lançamentos dos dados, o programa deve ser capaz de indicar, em qualquer momento do torneio, a classificação dos jogadores, em que casa se encontra cada jogador em jogo e, caso o torneio ainda não tenha terminado, quem é o próximo jogador a lançar os dados e se um dado jogador (em jogo) está, ou não, impedido de os lançar quando chegar a sua vez de jogar.



Figura 1: Tabuleiro do Jogo da Glória



Figura 2: Marcas

2 Regras do Torneio

Competem entre si de três a dez jogadores, cada um com uma marca diferente. Para simplificar, o jogador é identificado pela cor da sua marca e diz-se que está numa casa se a sua marca estiver nessa casa. Antes do torneio começar, é definida a ordem pela qual os jogadores jogam. Joga um jogador de cada vez. O jogador que joga vai rodando, respeitando a ordem previamente definida e considerando-se que depois do último vem o primeiro.

Por exemplo, se os jogadores tiverem marcas com as cores *A*, *B*, *C* e *D* (*amber*, *blue*, *charcoal* e *dark chocolate*) e for essa a ordem pela qual jogam: o primeiro a jogar é o jogador (que tem a marca) *A*; depois de *A* jogar, é a vez de *B* jogar; depois de *B* jogar, é a vez de *C* jogar; depois de *C* jogar, é a vez de *D* jogar; depois de *D* jogar, é novamente a vez de *A* jogar.

O torneio comprehende $J - 1$ jogos, se J for o número de jogadores. No início do torneio, todos os jogadores estão *em jogo* (nenhum jogador foi eliminado), pelo que todos os jogadores participam no primeiro jogo. Mas, no decurso ou no fim de cada jogo, um dos jogadores é *eliminado* do torneio (nunca mais joga) e só os jogadores que nunca foram eliminados permanecem em jogo, participando no jogo seguinte (se existir). O jogo em que só participam dois jogadores é o último. O torneio termina quando o último jogo termina: o vencedor desse jogo vence o torneio e o outro jogador é eliminado.

A eliminação de um jogador não altera a ordem pela qual os outros jogadores jogam. Se a ordem inicial fosse *A*, *B*, *C* e *D* (como no exemplo acima) e *B* fosse eliminado no fim do primeiro jogo, o segundo jogo seria jogado por *A*, *C* e *D*, que jogariam por esta ordem. O primeiro jogador a jogar seria *A*, depois seria *C*, depois seria *D*, depois seria *A*, etc. Quando um jogo começa, o primeiro jogador a jogar é sempre o primeiro na ordem previamente definida que ainda estiver em jogo.

O tabuleiro (usado em todos os jogos do torneio) tem um número C de casas, numeradas de 1 a C (com $10 \leq C \leq 150$). Qualquer casa tem um, e um só, tipo. Esse tipo pode ser *normal*, *pássaro*, *multa* ou *precipício*, existindo três variantes de precipícios: *caranguejo*, *inferno* e *morte*. Cada multa tem uma pena associada (um inteiro entre 1 e 4), que define durante quantas jogadas um jogador que pare na casa está impedido de lançar os dados. Pássaros, multas e precipícios só podem ocorrer entre as casas 2 e $C - 1$. Nesse intervalo, todas as casas “múltiplas de 9” (as casas 9, 18, 27, etc., sem ultrapassar $C - 1$) têm pássaro.

Quando um jogador está numa casa c , lança os dois dados e saem p pintas, o jogador *avança p casas* (onde $1 \leq c < C$ e $2 \leq p \leq 12$). Isto significa que o jogador passa a estar na casa $c + p$, exceto se essa casa não existir (por ser maior do que C), caso em que o jogador fica na casa C . De forma semelhante, um jogador que esteja numa casa c e que *recue p casas* passa a estar na casa $c - p$, exceto se essa casa não existir (por ser menor do que 1); nesse caso, o jogador vai para a casa 1.

No início de cada jogo, todos os jogadores (em jogo) estão na casa 1 e têm uma pena de zero jogadas. Quando chega a sua vez de jogar, o comportamento do jogador depende do valor da sua pena. A jogada pode alterar a casa onde está, a pena que tem e o seu estado (que poderá passar a eliminado). Na descrição que se segue, se não se explicitar que o valor da pena muda, a pena do jogador não é alterada e, se não se afirmar que o jogador é eliminado, o jogador permanece em jogo.

- Se o valor da pena for positivo, o jogador está impedido de lançar os dados (tendo de os passar ao “parceiro do lado”). O jogador permanece na mesma casa e o valor da sua pena baixa uma unidade. Considera-se que **estas jogadas ocorrem assim que é possível**.
- Se o valor da pena for zero, o jogador lança os dados. Se for a primeira jogada do jogador no jogo e saíram 6 pintas num dos dados e 3 pintas no outro, a marca do jogador vai para a casa C . Nos restantes casos, a sua marca avança tantas casas quanto o número total de pintas que saíram e, nas cinco seguintes situações, realizam-se as operações descritas:
 - Se a marca ficar num pássaro, o jogador avança (mais) 9 casas.

- Se a marca ficar numa multa, a pena do jogador aumenta entre 1 e 4 unidades, dependendo do valor que estiver especificado no tabuleiro para essa casa.
- Se a marca ficar num precipício-caranguejo, o jogador volta à casa onde estava (antes de ter lançado os dados) e depois recua tantas casas quanto o número total de pintas que saíram.
- Se a marca ficar num precipício-inferno, o jogador vai para a casa 1.
- Se a marca ficar num precipício-morte e for a primeira vez no jogo que uma marca para num precipício-morte, o jogador é imediatamente eliminado do torneio.¹

Regra geral, quando um jogador é eliminado por ter parado num precipício-morte, o jogo continua com os outros jogadores (quase como se a eliminação não tivesse ocorrido). É claro que o jogador eliminado nunca mais joga (nem no jogo em que participava, nem no torneio). No entanto, se ficar um único jogador em jogo, o jogo termina, porque esse jogador é de certeza o vencedor do jogo (e do torneio). Em todos os outros casos, um jogo só termina quando um dos jogadores chegar à casa C . Esse jogador é o que ganha o jogo. Nesse momento, se nenhum jogador tiver sido eliminado durante o jogo, um dos jogadores é eliminado e o jogo seguinte começa com menos um participante, exceto se só ficou um jogador em jogo, caso em que o torneio termina. A eliminação do jogador, se existir, e o começo do jogo seguinte ou a terminação do torneio ocorrem imediatamente a seguir ao jogo terminar. Depois do torneio terminar, o vencedor não pode jogar (e a sua marca permanece na casa onde estava quando o último jogo acabou).

Se ninguém tiver sido eliminado durante o jogo, o jogador eliminado no final do jogo é aquele que estiver na casa menor (mais longe da casa C) no momento em que o jogo termina; se houver vários jogadores nessa casa, o jogador eliminado é, de entre eles, o último a jogar na ordem previamente definida.

Quando o torneio termina, a classificação ordena os jogadores pela ordem inversa à da eliminação. O último classificado é o primeiro jogador eliminado (o que foi eliminado no decurso ou no fim do primeiro jogo), o penúltimo classificado é o segundo jogador eliminado, o antepenúltimo classificado é o terceiro jogador eliminado, e assim sucessivamente. O primeiro classificado é o vencedor do torneio, que nunca é eliminado. Enquanto o torneio decorre, há vários jogadores em jogo que, na classificação, ocorrem acima dos jogadores já eliminados. Um jogador i , em jogo, está mais bem classificado do que outro, j , também em jogo, se i tiver ganho mais jogos do que j ; se ambos tiverem ganho o mesmo número de jogos, se i estiver numa casa superior à casa de j (i.e., se i estiver mais perto da casa C); se ambos tiverem ganho o mesmo número de jogos e estiverem na mesma casa, se i joga antes de j na ordem previamente definida.

3 Especificação do Sistema

Pretende-se que a interface da aplicação seja simples, para poder ser utilizada em ambientes diversos e permitir automatizar o processo de teste. Por estes motivos, a entrada e a saída têm de respeitar o formato preciso especificado nesta secção. Pode assumir que o input obedece às restrições de valor e de formato indicadas, ou seja, que o utilizador não comete erros, para além dos previstos neste enunciado.

O programa lê linhas da entrada padrão (`System.in`) e do ficheiro de texto com o nome `boards.txt`, guardado na diretoria corrente, escreve linhas na saída padrão (`System.out`) e distingue maiúsculas de minúsculas (por exemplo, as palavras “exit” e “Exit” são diferentes).

Forma do Input

O ficheiro `boards.txt` tem a seguinte estrutura (onde k denota o número positivo de tabuleiros no ficheiro e o símbolo \leftrightarrow representa uma mudança de linha):

¹Se a marca ficar num precipício-morte e alguma marca já tiver parado num precipício-morte durante o jogo, é como se o jogador tivesse ficado numa casa normal (entre 1 e $C - 1$).

```

 $C_1 \leftarrow$ 
 $m_1 \leftarrow$ 
 $multa_1 \ pena_1 \leftarrow$ 
 $multa_2 \ pena_2 \leftarrow$ 
 $\dots$ 
 $multa_{m_1} \ pena_{m_1} \leftarrow$ 
 $p_1 \leftarrow$ 
 $prec_1 \ tipo_1 \leftarrow$ 
 $prec_2 \ tipo_2 \leftarrow$ 
 $\dots$ 
 $prec_{p_1} \ tipo_{p_1} \leftarrow$ 
 $C_2 \leftarrow$ 
 $m_2 \leftarrow$ 
 $multa_1 \ pena_1 \leftarrow$ 
 $multa_2 \ pena_2 \leftarrow$ 
 $\dots$ 
 $multa_{m_2} \ pena_{m_2} \leftarrow$ 
 $p_2 \leftarrow$ 
 $prec_1 \ tipo_1 \leftarrow$ 
 $prec_2 \ tipo_2 \leftarrow$ 
 $\dots$ 
 $prec_{p_2} \ tipo_{p_2} \leftarrow$ 
 $\dots$ 
 $C_k \leftarrow$ 
 $m_k \leftarrow$ 
 $multa_1 \ pena_1 \leftarrow$ 
 $multa_2 \ pena_2 \leftarrow$ 
 $\dots$ 
 $multa_{m_k} \ pena_{m_k} \leftarrow$ 
 $p_k \leftarrow$ 
 $prec_1 \ tipo_1 \leftarrow$ 
 $prec_2 \ tipo_2 \leftarrow$ 
 $\dots$ 
 $prec_{p_k} \ tipo_{p_k} \leftarrow$ 

```

onde, para cada tabuleiro i (com $i = 1, 2, \dots, k$):

- C_i é um número inteiro entre 10 e 150;
- m_i e p_i são números inteiros entre 1 e um terço de C_i ;
- $multa_1, \dots, multa_{m_i}$ são m_i números inteiros por ordem crescente;
- $pena_1, \dots, pena_{m_i}$ são m_i números inteiros entre 1 e 4;
- $prec_1, \dots, prec_{p_i}$ são p_i números inteiros por ordem crescente;
- $tipo_1, \dots, tipo_{p_i}$ são p_i palavras, sendo cada uma “crab”, “death” ou “hell”;
- os números $multa_1, \dots, multa_{m_i}, prec_1, \dots, prec_{p_i}$ variam entre 2 e $C_i - 1$, são todos distintos e nenhum é múltiplo de 9.

Os tabuleiros aparecem de seguida (sem linhas em branco a separá-los).

A descrição de um tabuleiro começa com o número de casas do tabuleiro (C_i). Depois vem o número de multas (m_i). Seguem-se m_i linhas, cada uma com uma casa onde se aplica multa e com o valor da respectiva pena, separados por um espaço. Na linha seguinte está o número de precipícios (p_i). Em cada uma das restantes p_i linhas há informação sobre um precipício: a casa onde ocorre e o seu tipo, separados por um espaço. Se o tipo for “crab”, a casa é um precipício-caranguejo; se for “death”, é um precipício-morte; se for “hell”, é um precipício-inferno.

A ordem pela qual os tabuleiros ocorrem no ficheiro **boards.txt** vai ser usada para identificar (na entrada padrão) o tabuleiro utilizado no torneio: o primeiro tabuleiro no ficheiro é identificado pelo número um, o segundo tabuleiro no ficheiro é o número dois, e assim sucessivamente.

A entrada padrão tem a seguinte estrutura:

```

cores ←
nrTabuleiro ←
comando ←
comando ←
...
comando ←
exit ←

```

onde:

- **cores** é uma sequência de letras maiúsculas todas diferentes (de ’A’ a ’Z’), de comprimento entre 3 e 10;

- *nrTabuleiro* é um inteiro positivo que não excede o número de tabuleiros no ficheiro `boards.txt`;
- *comando* é um de cinco comandos (chamados *comando-jogador*, *comando-casa*, *comando-estado*, *comando-classificação* e *comando-lançamento*) ou um comando inválido, explicados a seguir.

A primeira linha da entrada padrão especifica quantos jogadores participam no torneio, quais são as suas cores e a ordem por que jogam. A segunda linha identifica o tabuleiro usado no torneio, através do número de ordem desse tabuleiro no ficheiro `boards.txt`. Segue-se um número arbitrário de comandos. A última linha tem um comando especial, o *comando-sair*, que só pode ocorrer na última linha porque faz terminar a execução do programa.

Comando-Jogador

O comando-jogador indica que se pretende saber quem é o próximo jogador a lançar os dados, no momento corrente do torneio. Este comando não altera o estado do torneio. As linhas com comandos-jogador têm:

`player`↔

O programa escreve uma linha na consola, distinguindo dois casos:

- Se o torneio já tiver terminado, a linha tem:

`The cup is over`↔

- Nos restantes casos, a linha tem a seguinte forma, onde *corJogador* representa a cor do próximo jogador a lançar os dados:

`Next to play: corJogador`↔

Comando-Casa

O comando-casa indica que se pretende saber em que casa está o jogador com a cor dada, no momento corrente do torneio. Este comando não altera o estado do torneio. As linhas com comandos-casa têm a seguinte forma (com um espaço a separar as duas componentes):

`square corJogador`↔

onde:

- *corJogador* é uma sequência não vazia de caracteres.

O programa escreve uma linha na consola, distinguindo três casos:

- Se *corJogador* não for a cor de um dos jogadores, a linha tem:

`Nonexistent player`↔

- Se *corJogador* for a cor de um jogador eliminado, a linha tem:

`Eliminated player`↔

- Nos restantes casos, a linha tem a seguinte forma, onde *casaJogador* representa a casa onde o jogador referido no comando se encontra:

`corJogador is on square casaJogador`↔

Comando-Estado

O comando-estado indica que se pretende saber se, no momento corrente do torneio, o jogador com a cor dada pode, ou não, lançar os dados, quando (e se) chegar a sua vez de jogar. Este comando não altera o estado do torneio. As linhas com comandos-estado têm a seguinte forma (com um espaço a separar as duas componentes):

status *corJogador*↔

onde:

- *corJogador* é uma sequência não vazia de caracteres.

O programa escreve uma linha na consola, distinguindo cinco casos:

- Se *corJogador* não for a cor de um dos jogadores, a linha tem:

Nonexistent *player*↔

- Se *corJogador* for a cor de um jogador e o torneio já tiver terminado, a linha tem:

The cup is over↔

- Se o torneio ainda não tiver terminado e *corJogador* for a cor de um jogador eliminado, a linha tem:

Eliminated *player*↔

- Se o torneio ainda não tiver terminado, *corJogador* for a cor de um jogador em jogo e esse jogador puder lançar os dados, quando e se chegar a sua vez de jogar, a linha tem a seguinte forma:

***corJogador* can roll the dice**↔

- Nos restantes casos, a linha tem a seguinte forma:

***corJogador* cannot roll the dice**↔

Comando-Classificação

O comando-classificação indica que se pretende saber a classificação dos jogadores, no momento corrente do torneio. Este comando não altera o estado do torneio. As linhas com comandos-classificação têm:

ranking↔

O programa escreve tantas linhas quantos os jogadores, cada uma com a informação de um jogador diferente. Considere que *corJogador* e *nrJogosGanhos* representam, respetivamente, a cor e o número de jogos ganhos de um jogador. Caso o jogador esteja em jogo, *casaJogador* denota a casa onde se encontra. A forma da linha depende do estado do jogador, no momento corrente do torneio.

- Se o jogador estiver em jogo, a linha tem a seguinte forma:

***corJogador*: *nrJogosGanhos* games won; on square *casaJogador*.**↔

- Se o jogador já tiver sido eliminado, a linha tem a seguinte forma:

corJogador: nrJogosGanhos games won; eliminated. ↵

Os jogadores em jogo ocorrem antes dos jogadores eliminados. As linhas referentes aos jogadores em jogo são escritas por ordem decrescente de número de jogos ganhos; em caso de empate no número de jogos ganhos, por ordem decrescente de casa; em caso de empate no número de jogos ganhos e na casa, pela ordem em que jogam. As linhas referentes aos jogadores eliminados são escritas por ordem inversa à da eliminação.

Comando-Lançamento

O comando-lançamento indica que, no momento corrente do torneio, o jogador que tem direito a lançar os dados lançou os dados e quantas pintas saíram em cada dado. As linhas com comandos-lançamento têm a seguinte forma (com um espaço a separar componentes consecutivas):

dice pintasDado₁ pintasDado₂ ↵

onde:

- *pintasDado₁* e *pintasDado₂* são números inteiros.

O programa deve usar esta informação para atualizar o estado do torneio, mas não deve escrever qualquer resultado, exceto nos dois casos seguintes, nos quais o estado do torneio não muda e o programa escreve uma linha:

- Se *pintasDado₁* ou *pintasDado₂* não for um número entre 1 e 6, a linha tem:

Invalid dice ↵

- Se *pintasDado₁* e *pintasDado₂* forem números entre 1 e 6, mas o torneio já tiver terminado, a linha tem:

The cup is over ↵

Comando-Sair

O comando-sair indica que se pretende terminar a execução do programa. A linha com o comando-sair tem:

exit ↵

O programa termina, escrevendo uma linha na consola. Distinguem-se dois casos:

- Se o torneio ainda não tiver terminado, a linha tem:

The cup was not over yet... ↵

- Se o torneio já tiver terminado, a linha tem a seguinte forma, onde *corVencedor* denota a cor do jogador que ganhou o torneio:

corVencedor won the cup! ↵

Comandos Inválidos

Sempre que o utilizador escrever uma linha que não comece com as palavras “player”, “square”, “status”, “ranking”, “dice” ou “exit”, o estado do torneio não deve ser alterado e o programa deve escrever uma linha com:

```
Invalid command←
```

4 Exemplos

Apresentam-se alguns exemplos que assumem que o ficheiro `boards.txt` tem o conteúdo apresentado na [Figura 3](#).² A coluna da esquerda ilustra a interação: o input está escrito a azul e o output a preto. Todas as linhas do input e do output terminam com o símbolo de mudança de linha, que se omitiu para aumentar a legibilidade. A coluna da direita tem informação para o leitor do enunciado, servindo apenas para relembrar as regras descritas anteriormente. Nessa coluna, “lança” abrevia “lança os dados”, “C” abrevia “fica na casa” e “P” abrevia “fica com a pena”.

Note que o conjunto de exemplos é muito incompleto: há muitas situações que não são ilustradas e que podem ocorrer.

```
18
2
5 2
10 1
4
11 death
13 crab
15 hell
17 crab
25
3
7 1
17 4
19 2
3
6 hell
20 hell
24 death
33
1
11 3
5
4 death
6 death
13 crab
14 crab
19 hell
```

Figura 3: Conteúdo do ficheiro `boards.txt`

²No Mooshak, o conteúdo do ficheiro `boards.txt` é diferente.

Exemplo 1

RGB 3 jogadores: R, G e B (que jogam por esta ordem).
1 Primeiro tabuleiro: 18 casas, 2 multas e 4 precipícios.
dice 2 2 R lança; **C 5; P 2** (multa de 2 em 5).
dice 3 2 G lança; **C 6; P 0**.
dice 5 4 B lança; **C 10; P 1** (multa de 1 em 10). R joga; **P 1**.
ranking
B: 0 games won; on square 10.
G: 0 games won; on square 6.
R: 0 games won; on square 5.
status R
R cannot roll the dice
status G
G can roll the dice
status B
B cannot roll the dice
player
Next to play: G
square RED
Nonexistent player
dice 0 6
Invalid dice
dice 3 6 G lança; **C 1** (inferno em 15); **P 0**. B joga; **P 0**. R joga; **P 0**.
square G
G is on square 1
status R
R can roll the dice
status B
B can roll the dice
player
Next to play: G
ranking
B: 0 games won; on square 10.
R: 0 games won; on square 5.
G: 0 games won; on square 1.
dice 6 2 G lança; **C 18** (pássaro em 9). 1º jogo termina: G ganha; R eliminado.
ranking
G: 1 games won; on square 1.
B: 0 games won; on square 1.
R: 0 games won; eliminated.
dice 1 1 G lança; **C 3; P 0**.
dice 3 1 B lança; **C 5; P 2** (multa de 2 em 5).
dice 1 3 G lança; **C 7; P 0**. B joga; **P 1**.
player
Next to play: G
dice 6 1 G lança; **C 14; P 0**. B joga; **P 0**.
ranking
G: 1 games won; on square 14.
B: 0 games won; on square 5.
R: 0 games won; eliminated.

```

player
Next to play: G
rank
Invalid command
dice 2 1           G lança; C 11 (caranguejo em 17); P 0.
dice 5 6           B lança; C 16; P 0.
ranking
G: 1 games won; on square 11.
B: 0 games won; on square 16.
R: 0 games won; eliminated.
dice 3 3           G lança; C 5 (caranguejo em 17); P 0.
square G
G is on square 5
square R
Eliminated player
status R
Eliminated player
dice 2 2           B lança; C 18. 2º jogo e torneio terminam: B ganha; G eliminado.
ranking
B: 1 games won; on square 18.
G: 1 games won; eliminated.
R: 0 games won; eliminated.
square R
Eliminated player
square B
B is on square 18
square G
Eliminated player
square Blue sky
Nonexistent player
player
The cup is over
status B
The cup is over
status G
The cup is over
status dark chocolate
Nonexistent player
dice -1 -1
Invalid dice
Dice 7
Invalid command
dice 1 1
The cup is over
1000
Invalid command
exit
B won the cup!

```

Exemplo 2

ABCDE 5 jogadores: A, B, C, D e E (que jogam por esta ordem).
3 Terceiro tabuleiro: 33 casas, 1 multa e 5 precipícios.
dice 5 6 A lança; **C 12; P 0.**
dice 2 1 B lança; **C 4 (morte); é eliminado.**
ranking

A: 0 games won; on square 12.
C: 0 games won; on square 1.
D: 0 games won; on square 1.
E: 0 games won; on square 1.
B: 0 games won; eliminated.

square B Eliminated player
status B Eliminated player
dice 5 5 C lança; **C 11; P 3 (multa de 3 em 11).**
dice 3 2 D lança; **C 6 (morte); P 0.** Continua em jogo.
player
Next to play: E
dice 6 5 E lança; **C 12; P 0.**
ranking

A: 0 games won; on square 12.
E: 0 games won; on square 12.
C: 0 games won; on square 11.
D: 0 games won; on square 6.
B: 0 games won; eliminated.

dice 1 6 A lança; **C 1 (inferno em 19); P 0.** C joga; **P 2.**
dice 6 6 D lança; **C 27 (pássaro em 18); P 0.**
dice 1 1 E lança; **C 10 (caranguejo em 14); P 0.**
ranking

D: 0 games won; on square 27.
C: 0 games won; on square 11.
E: 0 games won; on square 10.
A: 0 games won; on square 1.
B: 0 games won; eliminated.

dice 4 2 A lança; **C 7; P 0.** C joga; **P 1.**
square A
A is on square 7
status C
C cannot roll the dice
dice 4 2 D lança; **C 33.** 1º jogo termina: D ganha.
ranking

D: 1 games won; on square 1.
A: 0 games won; on square 1.
C: 0 games won; on square 1.
E: 0 games won; on square 1.
B: 0 games won; eliminated.

dice 4 3 A lança; **C 8; P 0.**
dice 6 3 C lança; **C 33 (1ª jogada).** 2º jogo termina: C ganha; E eliminado.

ranking

C: 1 games won; on square 1.
D: 1 games won; on square 1.
A: 0 games won; on square 1.
E: 0 games won; eliminated.
B: 0 games won; eliminated.

dice 1 4

A lança; **C** 6 (morte); é eliminado.

player

Next to play: C

dice 2 6

C lança; **C** 18 (pássaro em 9); **P** 0.

ranking

C: 1 games won; on square 18.
D: 1 games won; on square 1.
A: 0 games won; eliminated.
E: 0 games won; eliminated.
B: 0 games won; eliminated.

dice 3 6

D lança; **C** 33 (1^a jogada). 3º jogo termina: D ganha.

player

Next to play: C

ranking

D: 2 games won; on square 1.
C: 1 games won; on square 1.
A: 0 games won; eliminated.
E: 0 games won; eliminated.
B: 0 games won; eliminated.

dice 1 2

C lança; **C** 4 (morte); é eliminado. 4º jogo termina: D ganha.

player

The cup is over

ranking

D: 3 games won; on square 1.
C: 1 games won; eliminated.
A: 0 games won; eliminated.
E: 0 games won; eliminated.
B: 0 games won; eliminated.

exit

D won the cup!

Exemplo 3

MNOABCXYZ

9 jogadores (que jogam pela ordem indicada).

2

Segundo tabuleiro: 25 casas, 3 multas e 3 precipícios.

player

Next to play: M

dices 0 0

Invalid command

status X

X can roll the dice

dice 3 2

M lança; **C** 1 (inferno em 6); **P** 0.

exit

The cup was not over yet...

5 Entrega do Trabalho

O trabalho é entregue no [Mooshak](#). Deverá submeter um arquivo `.zip` ao Problema A do concurso **IP2223-P2**. Não se esqueça que:

- O arquivo deve conter apenas todos os ficheiros `.java` que tiver criado para resolver o problema.
- O arquivo tem necessariamente de conter um ficheiro `Main.java`, onde está o método `main`.
- A classe `Main` tem de estar na raiz do arquivo (tem de pertencer ao pacote principal (`default`)).
- A versão de Java instalada no Mooshak é a 8.³

Cada grupo tem de se registar no concurso IP2223-P2, de acordo com as seguintes regras:

- O **nome do utilizador** (no Mooshak) tem de ter a forma `xxxxx-yyyyy`, onde `xxxxx` e `yyyyy` denotam os números de aluno dos membros do grupo.
- O **grupo** (no Mooshak) é o do respetivo **turno das aulas práticas**.⁴
- O **endereço de email** (para o qual o Mooshak envia a *password*) tem de ser o endereço institucional de um dos membros do grupo.

Por exemplo, o grupo constituído pelos alunos com os números 98765 e 98789, ambos inscritos no turno P5, é o utilizador com o nome `98765_98789` e pertence ao grupo P5. Só serão considerados entregues (e avaliados) os programas dos utilizadores no concurso IP2223-P2 que respeitem estas regras.

O concurso IP2223-P2 abre no dia 30 de novembro e encerra às **20h00** do dia **10 de dezembro de 2022** (sábado). Podem ressubmeter o trabalho as vezes que entenderem, até à hora limite de submissão. Apenas será avaliado o programa que obtiver a **maior pontuação** no Mooshak; se houver vários programas com a maior pontuação, será avaliado, de entre esses, o **último** que tiver sido submetido. Se quiserem que o programa avaliado seja outro, têm de enviar uma mensagem ao Professor Artur Miguel Dias (amd@fct.unl.pt) até uma hora após o concurso fechar, indicando o número da submissão que pretendem que seja avaliada.

6 Critérios de Avaliação do Trabalho

De acordo com o [Regulamento de Avaliação de Conhecimentos da FCT NOVA](#):

- Existe fraude quando:
 - (a) Se utiliza ou tenta utilizar, sob qualquer forma, num teste, exame, ou outra forma de avaliação, presencial ou a distância, informação ou equipamento não autorizado;
 - (b) Se presta ou recebe colaboração não autorizada na realização dos exames, testes, ou qualquer outra prova de avaliação de conhecimentos individuais;
 - (c) Se presta ou recebe colaboração, não permitida pelas regras aplicáveis a cada caso, na realização de trabalhos práticos, relatórios ou outros elementos de avaliação.

³Esta informação é irrelevante se só usar as instruções de Java dadas nas aulas porque, nesse caso, o programa deverá ter o mesmo comportamento na sua máquina e no Mooshak.

⁴Os alunos de turnos práticos diferentes que forem autorizados a fazer o trabalho em conjunto receberão instruções sobre o grupo do Mooshak que devem selecionar.

- Os estudantes diretamente envolvidos numa fraude são liminarmente reprovados na disciplina, sem prejuízo de eventual procedimento disciplinar ou cível.

Em IP, o documento [Colaboração Permitida e Não Permitida](#), disponibilizado no Moodle, clarifica as alíneas transcritas acima. Os alunos que cometerem fraude num trabalho não obterão frequência.

A avaliação do trabalho tem duas componentes independentes, cujas notas se somam para obter a nota do trabalho:

- **Funcionalidade** (correção dos resultados produzidos): **12 valores**

Um programa submetido ao concurso que só use as classes da biblioteca permitidas e que obtenha P pontos no Mooshak terá $P/10$ valores.⁵

As classes da biblioteca permitidas são **apenas as que foram usadas nas aulas teórico-práticas**.

- **Qualidade do código:** **8 valores⁶**

Um código com qualidade tem, entre outras, as seguintes características:

- **Várias classes que caracterizem bem as diferentes entidades do problema;**
- Classes, métodos, variáveis e constantes com objetivos bem definidos e as restrições de acesso apropriadas;
- Algoritmos simples e bem estruturados, implementados com as instruções mais adequadas;
- Identificadores que expressem os conceitos que representam, escritos de acordo com as convenções ensinadas (por exemplo, o nome de uma classe deve ser um substantivo que começa com uma letra maiúscula);
- Precondições nos métodos públicos;
- Indentação correta (lembre-se do comando CTRL-SHIFT-F do Eclipse), linhas com 100 caracteres (no máximo)⁷ e métodos com 25 linhas (no máximo);
- Um comentário no início de cada classe, que indique o que os objetos da classe representam, e um comentário antes de cada método, que explique resumidamente o que o método faz.

As **discussões** dos trabalhos são obrigatórias e decorrerão nos dias **12, 13 e 15 de dezembro de 2022**, durante as aulas (teórico-práticas ou práticas). Receberão mais informação até 11 de dezembro.

De acordo com os métodos de avaliação, escritos no CLIP desde o início do semestre, a nota de cada membro do grupo depende da nota do trabalho e do desempenho individual na discussão. Consequentemente, as notas dos dois elementos do grupo podem ser diferentes.

Bom trabalho!

⁵O trabalho pode ser realizado incrementalmente. Por exemplo, pode começar por assumir que nenhum jogador para num precipício-morte. É claro que, enquanto o programa não produzir os resultados corretos em todos os testes do Mooshak, não obterá 120 pontos.

⁶Note que a qualidade do código tem um peso muito grande na nota do trabalho.

⁷Na contagem do número de caracteres de uma linha de código, considere que um tab equivale a 4 caracteres.