10d 23h left

ALL

1

2

3

4

1. UAb E-fólio A, 2021, Alínea A

O e-fólio A é constituído por 4 alíneas, valendo 1 valor cada, devendo as mesmas serem realizadas sequencialmente, e podendo ser reutilizado código entre alíneas. A cotação total do e-fólio é de 4 valores. Os critérios de correção encontram-se no espaço da UC, sendo que 50% é destinado à funcionalidade, resultante da percentagem de casos de teste corretos. A realização do e-fólio na plataforma HackerRank não dispensa a entrega do relatório no espaço da UC. Deve em qualquer caso submeter terminando a submissão no HackerRank, de modo a poder ser apurado o critério da funcionalidade.

O relatório deve indicar as alíneas realizadas e resultados obtidos, e descrever o código realizado e opções tomadas, o qual não deve ultrapassar as 4 páginas. Se realizou parcialmente uma das alíneas, descreva o que fez e como planeava completar a alínea. Deve colocar o código das alíneas realizadas no anexo, mesmo as que foram realizadas parcialmente, e não colocar código no corpo do relatório.

O enunciado inclui uma história envolvendo um cliente, em que você é contratado(a) para desenvolver um programa. Pretende-se desta forma aproximar o e-fólio de uma situação real.

Enunciado:

Epaminondas é um empresário entusiasta de jogos e tem uma ideia fantástica sobre um novo pequeno jogo do tipo do xadrez e pretende alguém para as implementar. No entanto é muito desconfiado, e não quer revelar as suas ideias com medo que sejam roubadas. Apenas se tiver a certeza que é capaz de as implementar, é que as irá revelar à medida que desenvolve o programa. Assim, deve começar por implementar o programa A, passando para o programa B, e assim sucessivamente, até chegar ao programa D. Mesmo o programa D encontra-se dividido em duas partes.

O programa A deve receber **uma casa** de um tabuleiro com 6 colunas de largura e 5 linhas de altura e verificar se a **casa é válida**. As casas devem estar identificadas pela coluna (uma letra de 'a' a 'f'), e pela linha (um número de '1' a '5'). Epaminondas dá-lhe acesso especial ao tabuleiro do misterioso jogo, com todas as casas identificadas:

a5	b5	c5	d5	e5	f5
a4	b4	c4	d4	e4	f4
a3	b3	c3	d3	e3	f3
a2	b2	c2	d2	e2	f2
a1	b1	c1	d1	e1	f1

Como estava preocupado que mesmo com esta explicação possa não ter percebido

Language: C

Autocomplete Ready

```c
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <limits.h>
#include <stdbool.h>

int main() {
    char casa[80];

    scanf("%s", casa);




    return 0;
}
```

Line: 19 Col: 2

Test Results | Custom Input | Run | Submit Code

2. UAb E-fóio A, 2021, Alínea B

Epaminondas ficou bastante satisfeito com o seu programa A e pretende agora que o programa B possa mostrar o tabuleiro com uma casa selecionada. Para tal, pretende identificar a numeração das linhas e colunas da seguinte forma:

```
5:......
4:......
3:......
2:......
1:......
  abcdef
```

Na casa selecionada deve aparecer um 'X'. Caso a casa seja inválida, a saída deverá ser igual ao programa A, indicando que a casa é inválida.

Preocupado que possa ser mal interpretado, sendo a questão do tabuleiro muito importante, Epaminondas elabora também para o programa B um conjunto de casos de teste com o que pretende:

Entrada	Saída

c3	`5:......`
	`4:......`
	`3:..X...`
	`2:......`
	`1:......`
	`  abcdef`

a5	`5:X.....` `4:......`

Language: C

Autocomplete Ready

```c
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <limits.h>
#include <stdbool.h>

int main() {



    return 0;
}
```

Line: 17 Col: 2

Test Results | Custom Input | Run | Submit Code

10d 23h left

ALL

1

2

3

4

3. UAb E-fóio A, 2021, Alínea C

A sua resolução do programa B deixou Epaminondas mais entusiasmado, e quer agora passar a outra fase. É tempo de revelar as peças do jogo. Pretende colocar no tabuleiro as peças para qualquer posição de jogo.

Existem dois tipos de peças, o **Rei** e o **Peão**, e dois lados, as **brancas** e as **pretas**. Cada peça pode ser das brancas ou das pretas, existindo inicialmente **um rei** e **6 peões** para cada lado. No decorrer do jogo, cada jogador efetua a sua jogada alternadamenet, sendo os movimentos detalhados apenas numa fase mais avançada da implementação. No entanto há peças que podem ir sendo tomadas, pelo que numa posição podem existir menos de 6 peões. O rei é que não pode ser tomado, na verdade, quem tomar o rei adversário, ganha, e o jogo acaba. Portanto, não há posições válidas sem o rei para cada lado, mas existem posições válidas com menos de 6 peões. Cada lado pode ter entre 0 e 6 peões.

Em vez do X utilizado no programa B, Epaminondas pretende que utilize as seguintes letras para as peças de cada lado:

	Brancas	Pretas
Rei	R	r
Peão	P	p

Como há dois reis e um número entre 0 e 6 peões para cada lado, Epaminondas pretende que o programa C aceite em vez de uma só casa, duas casas com a posição de cada rei (primeiro o rei branco seguido do rei preto), e 6 casas para os peões brancos, seguidos de 6 casas para os peões pretos. Caso as casas dos reis sejam inválidas, a posição é inválida, mas se uma das casas de peões for inválida, deve-se considerar simplesmente que o peão não existe. Considera-se também que não serão dadas casas iguais, pelo que não há necessidade de considerar inválida uma posição que tenha uma casa repetida.

Epaminondas preparou a seguinte lista com a ordem pela qual devem ser introduzidas as casas, utilizando a letra da peça para identificar a peça que deve ser carregada para cada

Language: C

Autocomplete Ready

```c
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <limits.h>
#include <stdbool.h>

int main() {



    return 0;
}
```

Line: 17 Col: 2

Test Results | Custom Input | Run | Submit Code

Epaminondas preparou a seguinte lista com a ordem pela qual devem ser introduzidas as casas, utilizando a letra da peça para identificar a peça que deve ser carregada para cada casa:

R	r	P	P	P	P	P	P	p	p	p	p	p	p

Atendendo a que a possibilidade de carregar qualquer posição válida, e mostrar no computador, seria um passo muito importante para a implementação do jogo, Epaminondas não deixou de preparar um conjunto de casos de teste com o pretendido, neste tão delicado passo.

Entrada	Saída
d1 d5 a2 b2 c2 d2 e2 f2 a4 b4 c4 d4 e4 f4	5:...r.. 4:pppppp 3:...... 2:PPPPPP 1:...R.. abcdef
d1 d5 a2 b2 0 d2 e2 0 a4 b4 0 d4 e4 0	5:...r.. 4:pp.pp. 3:...... 2:PP.PP. 1:...R.. abcdef
d3 b5 a3 b2 0 0 e2 0 a4 b3 0 d4 0 0	5:.r.... 4:p..p.. 3:Pp.R.. 2:.P..P. 1:...... abcdef

Language: C

Autocomplete Ready

```c
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <limits.h>
#include <stdbool.h>

int main() {



    return 0;
}
```

Line: 17 Col: 2

Test Results | Custom Input | Run | Submit Code

10d 23h left

ALL

1

2

3

4

	3:pP.... 2:P.P... 1:..... abcdef
d4 a5 a2 b3 X X X X X a3 b4 X X X	5:r..... 4:.p.R.. 3:pP.... 2:P..... 1:..... abcdef
f1 f5 a2 b3 c2 d3 e2 f3 a3 b4 c3 d4 e3 f4	5:.....r 4:.p.p.p 3:pPpPpP 2:P.P.P. 1:.....R abcdef
a1 f5 a2 b3 X X e2 f3 a3 b4 X X e3 f4	5:.....r 4:.p...p 3:pP..pP 2:P...P. 1:R..... abcdef

Embora não esteja nos casos de teste, se uma das duas primeiras casas não for válida, o programa deve simplesmente devolver:

```
Posicao invalida.
```

Language: C

Autocomplete Ready

```c
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <limits.h>
#include <stdbool.h>

int main() {



    return 0;
}
```

Line: 17 Col: 2

Test Results | Custom Input | Run | Submit Code

10d 23h left

ALL

4. UAb E-fóio A, 2021, Alínea D

O seu trabalho até aqui leva Epaminondas a acreditar que é possível implementar o seu misterioso jogo, e está pronto a revelar-lhe duas partes essenciais: os movimentos das peças; um conjunto de regras para implementar um jogador automático.

Seria fantástico se conseguir fazer ambas as partes, mas Epaminondas ficaria já bastante satisfeito se conseguir implementar os movimentos das peças, pelo que divide o programa D em duas partes, ficando metade dos casos de teste em cada parte.

Movimento das peças:

A entrada de dados do programa D é inicialmente igual ao programa C, mas seguida de movimentos de peças. Cada lado joga à vez, correspondendo um lance à indicação de duas casas válidas, casa de origem e casa de destino, que correspondam a um lance válido. Caso exista uma casa inválida ou um lance inválido, o programa deve parar imediatamente o jogo, com vitória para o último jogador que fez um lance válido. Caso a posição fique sem um dos reis, ganha o último jogador a efetuar o lance, neste caso de tomada do rei adversário.

Os lances válidas são os que têm casa de origem e de destino válidas, com as seguintes regras:

Situação	Casa de origem	Casa de destino
Conteúdo das casas	Contém peça do jogador atual	Tem de estar vazia ou ter uma peça adversária
Movimento do Rei	Rei do jogador atual	Tem de estar em contacto com a casa de origem (podem haver diagonais)
Movimento de Peão	Peão do jogador atual	Tem de estar vazia, e logo a seguir ao peão (linha+1 para as brancas, e linha-1 para as pretas)
Tomada de Peão	Peão do jogador atual	Tem de ter uma peça adversária (linha+1 e coluna +/-1 para as brancas, e linha-1 e coluna +/- 1 para as pretas)

Estas regras Epaminondas reconhece serem complicadas, pelo que faz os seguintes esquemas de exemplo:

Language: C

Autocomplete Ready

```c
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <limits.h>
#include <stdbool.h>

int main() {



    return 0;
}
```

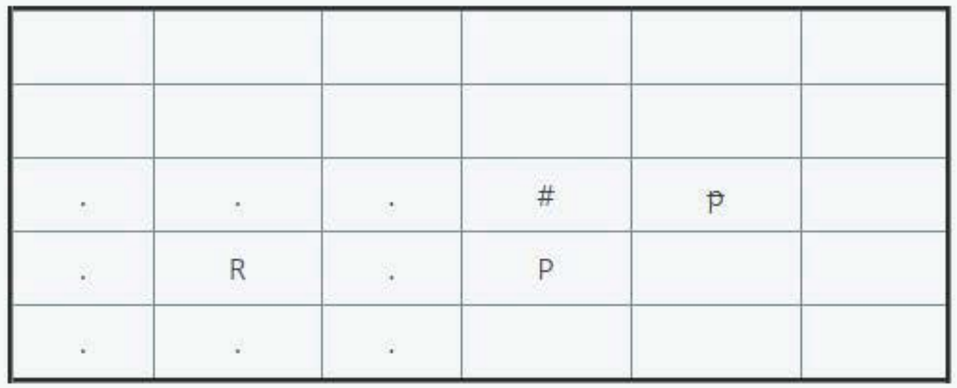
Line: 14 Col: 2

Test Results | Custom Input | Run | Submit Code

10d 23h left

ALL

Estas regras Epaminondas reconhece serem complicadas, pelo que faz os seguintes esquemas de exemplo:



O rei branco pode mover para qualquer casa adjacente, indicadas com um '.'. O peão pode mover para a frente, a casa marcada com '#' e pode tomar o peão marcado com 'p'. Poderia ter tomado para o outro lado. No caso dos peões pretos, a direção inverte-se, movendo-se os peões para baixo no tabuleiro.

Em termos de output, Epaminondas pretende que seja mostrado o tabuleiro final, e quem ganhou, bem como o número de jogadas válidas. Atendendo a que já é tarefa complicada, disponibiliza desde já um primeiro conjunto de casos de teste para validar partidas, sendo destacado a vermelho o lance inválido:

Entrada	Saída

d1 d5 a2 b2 c2 d2
e2 f2 a4 b4 c4 d4
e4 f4 c2 c3 d4 c3
c3 c4
	`5:...r..`
	`4:ppp.pp`
	`3:..p...`
	`2:PP.PPP`
	`1:...R..`
	`abcdef`
	`Ganham as pretas. Partida com 2 jogadas validas.`

d1 d5 a2 b2 c2 d2
e2 f2 a4 b4 c4 d4
e4 f4 c2 c3 d4 c3
d1 c2 c3 b2 c2 b2
X
	`5:...r..`
	`4:ppp.pp`
	`3:......`
	`2:PR.PPP`
	`1:`

Language: C

Autocomplete Ready

```c
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <limits.h>
#include <stdbool.h>

int main() {



    return 0;
}
```

Line: 14 Col: 2

Test Results | Custom Input | Run | Submit Code

10d 23h left

ALL

1

2

3

4

Epaminondas realça que é importante passar estes casos de teste antes de avançar para a implementação do jogador automático. Naturalmente se não conseguir a implementação do jogador automático, a validação das partidas é já um bom resultado.

Jogador Automático:

Epaminondas pretende agora que o computador possa também fazer uma jogada. Para tal, em vez de um lance com duas casas, deve ser colocado JA (Jogador Automático), e nesse caso, em vez de se realizar um lance, é o computador que executa um lance de acordo com um conjunto de regras selecionadas. Essas regras foram fruto de muito estudo por parte de Epaminondas, que pretende agora que as implemente de modo a obter um jogador automático de grande nível. Para tal, Epaminondas divide as jogadas em 5 categorias, e existindo pelo menos um lance de uma categoria, a categoria seguinte não é considerada. As categorias são o seguinte:

Nível	Categoria	Descrição
1	Tomar Rei	Lances em que se toma o rei adversário.
2	Tomar Peões	Lances em que peões tomam peões. No caso de não existirem lances deste tipo, inclui-se aqui lances em que o rei toma um peão desprotegido.
3	Mover Peões	Lances em que os peões avançam para a frente, sem tomar peças.
4	Avançar o Rei	Avançar o rei em direção ao adversário, no caso da distância entre reis for maior que 2. Deve-se utilizar a direção mais distante (linhas/colunas). Em caso de igualdade avançar nas colunas. Se estas condições se verificarem, considera mover nas 3 casas dessa direção (pode avançar e ir para os lados também), mas destas três casas desconsiderar casas inválidas e casas que estejam atacadas por peões adversários.
5	Mover o Rei	Em último caso, considerar todos os movimentos válidos do rei.

Após obter a lista de lances de determinado nível, no caso de existir um só lance, executar o lance. Caso contrário selecionar o lance que tiver como casa de destino o mais para cima possível (inverso para as pretas). No caso de empate, considerar a casa de destino mais

Language: C

Autocomplete Ready

```c
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <limits.h>
#include <stdbool.h>

int main() {



    return 0;
}
```

Line: 14 Col: 2

Test Results | Custom Input | Run | Submit Code