# ☆ UAb E-fólio A, 1819, 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 50% da nota destinado à avaliação da funcionalidade, resultante da percentagem de casos de teste corretos. A realização do e-fólio deve ser feita na plataforma *HackerRank*, sendo válidas apenas as submissões dadas como terminadas, e não dispensando a entrega do relatório no espaço da UC.

O relatório deve indicar as alíneas realizadas e resultados obtidos, e descrever o código realizado e opções tomadas, e 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.

Este e-fólio é baseado no Dominó, mas uma versão em que as peças têm 4 números em vez de dois, e são quadradas. Exemplo de uma peça:

| | |
|---|---|
| 3 | 0 |
| 2 | 2 |

Esta peça pode rodar, podendo neste caso ficar nas seguintes três posições alternativas:

| | |
|---|---|
| 0 | 2 |
| 3 | 2 |

(rotação de 1 no sentido anti-horário da peça exemplo)

| | |
|---|---|
| 2 | 2 |
| 0 | 3 |

(rotação de 2 no sentido anti-horário da peça exemplo)

| | |
|---|---|
| 2 | 3 |

1

2

3

4

Tal como as peças de dominó clássicas, estas peças têm costas todas iguais de modo a poderem ser baralhadas, pelo que não se podem virar, apenas rodar. Ou seja, a peça a cima nunca pode estar na seguinte posição:

| | |
|---|---|
| 2 | 0 |
| 2 | 3 |

Os números possíveis podem ser valores entre 0 e 9. As diversas alíneas do e-fólio irão relevar a cada momento o jogo/puzzle proposto com este conjunto de peças.

Pretende-se na alínea A que receba 4 valores inteiros A B C D (de 0 a 9) e mostre a peça respeitando a seguinte ordem:

| | |
|---|---|
| A | B |
| D | C |

Caso de teste 1:

**Entrada:**

```
3 0 2 2
```

**Saída:**

```
++++
+30+
+22+
++++
```

Caso de teste 2:

**Entrada:**

```
2 3 0 2
```

**Saída:**

```
++++
+23+
```

Notar que a borda da peça é feita com o operador de soma: '+'

**YOUR ANSWER**

1

2

3

4

We recommend you take a quick tour of our editor before you proceed. The timer will pause up to 90 seconds for the tour. Start tour

Original code | C

```
1  #include <math.h>
2  #include <stdio.h>
3  #include <string.h>
4  #include <stdlib.h>
5  #include <assert.h>
6  #include <limits.h>
7  #include <stdbool.h>
8
9  int main() {
10
11
12
13 }
```

Line: 1 Col: 1

Test against custom input

Run Code | Submit code & Continue

(You can submit any number of times)

Download sample test cases *The input/output files have Unix line endings. Do not use Notepad to edit them on windows.*

1

2

3

4

# ☆ UAb E-fólio A, 1819, Alínea B

Pretende-se na alínea B listar todas as peças com números de 0 a K, sendo K um valor inteiro entre 0 e 9 inclusive, e em todas as posições (o conceito de rotação não é relevante aqui). Dado que o número de peças e posições pode ser muito elevado, deve parar após um valor do número de peças W, fornecido pelo utilizador. No caso de utilizar vetores em que precise de um valor máximo para o número de peças, utilize o valor 4000.

Considerando as posições A B C D em baixo, pretende-se que listar primeiro todas as possibilidades de D antes de C, e assim sucessivamente até A. Ou seja, aparece primeiro a peça 0000 (considerando a ordem ABCD, de seguida a peça 0001, seguida da 0002, e no caso de K=2, então a próxima peça será a 0010.

| | |
|---|---|
| A | B |
| D | C |

Caso de teste 1:

**Entrada:**

```
2 6
```

**Saída:**

```
++++
+00+
+00+
++++
++++
+00+
+10+
++++
++++
+00+
+20+
++++
++++
+00+
+01+
++++
++++
```

```
+00+
+21+
++++
```

1

2

3

4

Este exemplo tem K=2 e W=6, sendo portanto mostradas as 6 primeiras peças, com números de 0 a 2. Notar que a segunda peça mostrada é igual à quarta peça. Tem apenas um número 1, e o resto 0, podendo ser rodada. Nesta alínea pretende-se que liste todas as peças e posições possíveis (não removendo peças repetidas considerando rotações).

**YOUR ANSWER**

We recommend you take a quick tour of our editor before you proceed. The timer will pause up to 90 seconds for the tour. Start tour

Original code | C

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

int main() {

}
```

Line: 1 Col: 1

Test against custom input

Run Code | Submit code & Continue

(You can submit any number of times)

Download sample test cases *The input/output files have Unix line endings. Do not use Notepad to edit them on windows.*

1

2

3

4

1

2

3

4

# ☆ UAb E-fólio A, 1819, Alínea C

As peças do Dominó clássico são únicas. Ou seja, não existem duas peças 2|3, apenas uma peça, podendo a mesma tomar a forma 2|3 ou 3|2 através da rotação. Pretende-se que implemente a unicidade das peças na alínea C, nesta versão de peças com 4 números. De todas as peças/posições identificadas na alínea B, pretende-se apenas as peças únicas, e numa posição normalizada, que é a primeira versão da peça que ocorre de acordo com a ordem utilizada na alínea B. Pretende-se ainda as peças por uma ordem fixa, que corresponde também à ordem utilizada na alínea B.

Nesta alínea pretende-se que remova a borda das peças, de modo a reduzir o volume de dados de saída, sendo os dados de entrada os mesmos dois parâmetros da alínea B, um valor K para o número mais alto, e um valor W para o número máximo de peças a visualizar. No final, pretende-se que indique o número de peças únicas geradas, independente do número de peças visualizadas.

Caso de teste 1:
**Entrada:**

```
1 100
```

**Saída:**

```
00
00

00
10

00
11

01
10

01
11

11
11
```

Notar que a segunda peça tem apenas um valor 1 e o resto zero. É gerada apenas esta versão da peça, não sendo gerada das restantes versões.
A ordem das peças segue a indicada na alínea B, mas após subtrair as versões que são iguais considerando operações de rotação.

1

Caso de teste 2:

**Entrada:**

2

```
2 6
```

3

**Saída:**

4

```
00
00

00
10

00
20

00
11

00
21

00
12

24 pecas geradas.
```

Notar que para o caso de K=1 existem 6 peças distintas, enquanto que com K=2 existem 24 peças distintas. Na alínea anterior, em que se mostrava todas as peças/posições distintas, existem K^4 peças/posições distintas, enquanto que ao considerar apenas peças únicas, este valor é naturalmente inferior.

**YOUR ANSWER**

1

2

3

4

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

int main() {


}
```

Line: 1 Col: 1

Test against custom input

Run Code

Submit code & Continue

(You can submit any number of times)

Download sample test cases *The input/output files have Unix line endings. Do not use Notepad to edit them on windows.*

1

2

3

4

# ☆ UAb E-fólio A, 1819, Alínea D

Pretende-se nesta alínea que selecione W peças, de entre as peças geradas na alínea anterior, que são as peças que tem para jogar. As peças devem ser baralhadas de acordo com a atividade formativa baralhar.c, a qual se descreve em baixo, devendo utilizar a função randaux de modo a poder reproduzir os casos de teste.

**Entrada:**

```
1 3
```

**Saída:**

```
6 pecas geradas.

0:
 01
 10
1:
 01
 11
2:
 11
 11
```

Nesta execução de exemplo (ainda incompleta), foi introduzido K=1 e W=3, significando que usamos peças de 0 a 1, e pretendemos extrair 3 peças aleatoriamente, de entre todas as peças únicas possíveis, as quais foram geradas na alínea C. Utilizando o método da função baralhar.c das atividades formativas, obtêm-se as três peças indicadas. A ordem a apresentar as peças selecionadas, respeita a ordem de geração das peças da alínea B, mesmo que tenham sido selecionadas pela função baralhar por outra ordem.

Pretende-se selecionar peças para colocar na mesa. A primeira peça pode ser colocada após rotação. As restantes peças podem ser colocadas após rotação mas apenas por baixo da peça mais abaixo da cadeia, à esquerda ou à direita desta, desde que o número de ligação seja igual em ambas as peças. Exemplo:

| | |
|---|---|
| 0 | 1 |
| 1 | **0** |

| | |
|---|---|
| 1 | 1 |

Estas duas peças, que correspondem ao ID 0 e 1, podem ser colocadas de forma ligada, em que a segunda fica à direita. A casa a azul tem o mesmo número, pelo que a jogada é válida. Esta peça poderia ter sido colocada também na esquerda:

| | |
|---|---|
| 0 | 1 |
| **1** | 0 |

| | |
|---|---|
| 0 | **1** |
| 1 | 1 |

O objetivo do puzzle é fazer uma sequência de jogadas válidas, de modo a gastar todas as peças, e assim terminar o puzzle com sucesso. Se fizer um lance ilegal, o puzzle termina com insucesso. Uma jogada é efetuada após serem inseridos três números inteiros: ID da peça (no caso do exemplo é um valor de 0 a 2), número de rotações (um valor de 0 a 3), e posição da peça (0 na esquerda, 1 na direita). Uma rotação corresponde a rodar a peça 90º no sentido anti-horário. Vamos continuar com alguns lances para exemplificar, colocando a azul os dados de entrada, e deixando a preto a saída.

```
1 3

6 pecas geradas.

0:
 01
 10
1:
 01
 11
2:
 11
 11
Mesa:
Jogada:0 0 0

0:
```

1
2
3
4

```
 11
Mesa:
01
10
Jogada:0 0 0

0:
 11
 11
Mesa:
 01
 10
01
11
Jogada:0 0 1

Mesa:
 01
 10
01
11
 11
 11
Puzzle terminado.
```

Notar que foi selecionada sempre a primeira peça em cada momento, portanto o primeiro dos três números foi sempre o número 0. O segundo número corresponde ao número de rotações a dar à peça, antes de a colocar na mesa, e foi sempre zero, portanto a posição foi a mesma. Apenas no último lance houve a escolha pela direita, e portanto o último número foi 1, mas neste caso teria sido válida a jogada de colocar a peça na esquerda.

Vamos agora dar um exemplo mais complexo, para ilustrar mais situações:

```
2 4

24 pecas geradas.

0:
 00
 10
1:
 00
 20
2:
```

1

2

3

4

```
 22
Mesa:
Jogada:2 0 0

0:
 00
 10
1:
 00
 20
2:
 12
 22
Mesa:
02
22
Jogada:2 3 1

0:
 00
 10
1:
 00
 20
Mesa:
02
22
 21
 22
Jogada:1 2 0

0:
 00
 10
Mesa:
02
22
 21
 22
02
00
Jogada:0 0 0

Mesa:
 02
 22
  21
  22
 02
```

Realça-se neste exemplo o facto de se utilizar várias rotações e colocar peças à esquerda, forçando que a cadeia de peças na mesa tenha de ser deslocada para a direita. Neste outro caso, exemplifica-se uma jogada ilegal, e respetiva mensagem de erro:

```
5 4

336 pecas geradas.

0:
 01
 12
1:
 01
 32
2:
 12
 22
3:
 34
 55
Mesa:
Jogada:0 0 0

0:
 01
 32
1:
 12
 22
2:
 34
 55
Mesa:
01
12
Jogada:0 0 0

0:
 12
 22
1:
 34
 55
Mesa:
 01
 12
```

```
Erro: jogada deve ser constituida por tres numeros: <ID> <rotacoes> <p
osicao>.
```

Neste exemplo a jogada foi sempre 0 0 0, o que corresponde a selecionar sempre a primeira peça, não a rodar e colocá-la na esquerda. No entanto, no terceiro lance, esta jogada é ilegal uma vez que o 3 não é igual ao 2. Para colocar a primeira peça teria-se de rodar a peça e colocá-la à direita, para fazer coincidir o número 2. Nesta situação retorna-se a mensagem de erro indicada: "Erro: jogada deve ser constituida por tres numeros: <ID> <rotacoes> <posicao>."

Existem casos de teste visiveis com uma jogada ilegal logo no início, de modo a permitir testar a extração dos números aleatórios, e para no caso de não ter a alínea completa, poder mesmo assim ficar com alguns casos de teste validados.

**Atividade Formativa: baralhar.c**

Complete um programa que utiliza uma função que recebe um vector e a sua dimensão, e baralhe os elementos nele contidos. Atenção que deve realizar o número mínimo de trocas, e a probabilidade de cada elemento estar em qualquer posição deve ser igual.

Notas:

- Para testar a função, faça um vetor identidade (na primeira posição tem o 0, na segunda o 1, e assim sucessivamente), e envie para a função baralhar;
- Começando pelo primeiro elemento, tem de haver um valor aleatório para decidir qual será o primeiro elemento. O segundo valor aleatório decidirá qual será o segundo elemento (não removendo o primeiro elemento que já foi escolhido aleatoriamente), e assim sucessivamente para todos os restantes elementos.

Exemplificação:

Na tabela abaixo, em cada linha está uma iteração, bem como o valor das diversas posições do vector, e o resultado de um valor aleatório gerado em cada iteração para decidir qual o elemento nessa iteração. O valor aleatório pertence a um intervalo cada vez mais pequeno, até que o último valor é um valor aleatório 0 ou 1, para decidir a ordem das duas últimas posições no vector.

```
It.|Valor Aleatório| 0 | 1 | 2 | 3
 0 |                | 48| 22| 80| 25
 1 |  rand()%4 = 2  |*80| 22|*48| 25
 2 |  rand()%3 = 0  | 80|*22| 48| 25
```

**Entrada**:

- Número de elementos (valor entre 10 e 1000)

**Saída**:

- 10 primeiros elementos baralhados

Execução de exemplo:

```
C:\>baralhar
1000
4
```

**YOUR ANSWER**

We recommend you take a quick tour of our editor before you proceed. The timer will pause up to 90 seconds for the tour. Start tour

Original code | C

```
    return(((seed = seed * 214013L + 2531011L) >> 16) & 0x7fff)
}
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <limits.h>
#include <stdbool.h>

int main() {


}
```

Line: 1 Col: 1

Test against custom input

Run Code | Submit code & Continue

(You can submit any number of times)

Download sample test cases *The input/output files have Unix line endings. Do not use Notepad to edit them on windows.*

1

2

3

4