

**MAC2166 – Introdução à Computação**  
ESCOLA POLITÉCNICA  
Prova Substitutiva – 30 de junho de 2015

Nome: \_\_\_\_\_

Assinatura: \_\_\_\_\_

Nº USP: \_\_\_\_\_ Turma: \_\_\_\_\_

Professor: \_\_\_\_\_

**Instruções:**

1. Não destaque as folhas deste caderno.
2. A prova consta de 2 questões. Verifique antes de começar a prova se o seu caderno de questões está completo.
3. Qualquer questão pode ser resolvida em qualquer página. Basta indicar SOLUÇÃO DA QUESTÃO X em letras ENORMES.
4. A prova pode ser feita a lápis. Cuidado com a legibilidade e, principalmente, com a TABULAÇÃO.
5. Não é necessário apagar rascunhos no caderno de questões.
6. Não é permitido o uso de folhas avulsas para rascunho.
7. Não é permitido o uso de equipamentos eletrônicos.
8. Não é permitido a consulta a livros, apontamentos ou colegas.
9. Você pode definir funções e métodos usá-los à vontade.

**DURAÇÃO DA PROVA: 2 horas**



Questão	Valor	Nota
1a	2,0	
1b	3,0	
2a	1,0	
2b	1,0	
2c	1,0	
2d	2,0	
Total	10,0	

Fonte: <http://geeksandglitter.wordpress.com/>

### QUESTÃO 1 (composta de 2 itens)

Esta questão consiste na implementação de duas funções em Python. Uma das funções é o `main()` e a outra é uma função de nome `lecker()`.

Dizemos que uma lista de números inteiros é **lecker** se ela tem **apenas 1** elemento que é **maior** que seus vizinhos. Por exemplo,

- `[2, 5, 10, 46, 25, 12, 7]` é lecker, pois 46 é único elemento que é maior que seus vizinhos, que são 10 e 25.
- `[13, 5, 4, 2, 3, 0, -3, -14]` **não é** lecker, porque 13 é maior que 5 (5 é o único vizinho do 13) e 3 é maior que 2 e 0 (2 e 0 são os vizinhos do 3);
- `[6, 7, 8, 9, 10]` é lecker, pois apenas 10 é maior que seus vizinhos (9 é o único vizinho do 10).
- `[7, 6]` é lecker, pois apenas 7 é maior que seus vizinhos (6 é o único vizinho de 7).
- `[16, 16, 16]` **não é** lecker, pois nenhum elemento é maior que seus vizinhos.
- `[23]` **não é** lecker, pois nenhum elemento é maior que seus vizinhos (23 não tem vizinhos).

#### Item (a) (vale 2,0 pontos)

Escreva uma função `lecker()` que **recebe** uma lista de números e retorna **True** se a lista é lecker e **False** em caso contrário.



Uma matriz é **lecker** se cada uma das listas formada por linhas da matriz e cada uma das listas formada por colunas da matriz forem lecker. Por exemplo,

numero de linhas = 3 numero de colunas = 4 matriz: 1 2 3 4 8 7 6 5 9 10 11 12 A matriz é lecker!	numero de linhas = 3 numero de colunas = 4 matriz: 1 2 3 5 8 7 6 4 9 10 11 12 A matriz não é lecker!	numero de linhas = 4 numero de colunas = 3 matriz: 1 2 3 5 8 7 6 4 9 10 11 12 A matriz não é lecker!	numero de linhas = 4 numero de colunas = 3 matriz: 1 2 3 5 8 7 6 9 13 10 11 12 A matriz é lecker!
--	--	---	--

### Item (b) (vale 3,0 pontos)

Escreva uma função `main()` que **leia** inteiros `n_lin` e `n_col`,  $0 < n\_lin, n\_col$ , e uma matriz de números inteiros com `n_lin` linhas e `n_col` colunas e **imprime** uma mensagem indicando se a matriz é ou não lecker. Utilize **obrigatoriamente** a função `lecker()` do item anterior mesmo que você não a tenha feito.

A seguir está um exemplo de execução da função `main()`. O seu programa pode utilizar **sem escrevê-la** uma função `imprima_matriz` que **recebe** uma matriz e a **imprime** como mostrado no exemplo de execução.

Verificador de leckerdesa

```
Digite o número de linhas da matriz: 3
Digite o número de colunas da matriz: 4
Digite o elemento [0][0]: 1
Digite o elemento [0][1]: 2
Digite o elemento [0][2]: 3
Digite o elemento [0][3]: 5
Digite o elemento [1][0]: 8
Digite o elemento [1][1]: 7
Digite o elemento [1][2]: 6
Digite o elemento [1][3]: 4
Digite o elemento [2][0]: 9
Digite o elemento [2][1]: 10
Digite o elemento [2][2]: 11
Digite o elemento [2][3]: 12
```

```
número de linhas = 3
número de colunas = 4
1 2 3 5
8 7 6 4
9 10 11 12
```

A matriz não é lecker!



## QUESTÃO 2 (composta de 4 itens)

Nesta questão você escreverá um programa em Python que simula uma partida de uma versão simplificação de um jogo de cartas chamado **War**<sup>1</sup>.

**War** é um jogo entre dois jogadores. No início de uma partida de **War** as cartas de um baralho são **embalhadas**. Em seguida, cada um dos dois jogadores **recebe** metade das 52 cartas do baralho. Cada carta **recebida** por um jogador é colocada em um **monte** à sua frente. Cada partida do jogo é composta por 26 rodadas. Em cada rodada, simultaneamente, cada jogador **pega a carta** que está no topo do seu monte e a mostra ao adversário. O **vencedor da rodada** é o jogador que tiver a carta de maior **valor**. Em uma rodada pode haver **empate**. O **vencedor da partida** é o jogador que vencer mais rodadas. Em uma partida também pode haver **empate**.

Nesta questão, os jogadores da partida de **War** simulada pelo seu programa serão dois físicos ilustres do Instituto de Tecnologia da Califórnia (Caltech): Sheldon Cooper (**Sheldon**) e Leonard Hofstadter (**Leonard**).

A seguir é exibido o resultado de uma partida de **War** jogada entre **Sheldon** e **Leonard**.

```
Sheldon: ♠K ♠Q ♠J ♦5 ♥3 ♦8 ♥2 ♠D ♦A ♣A ♦Q ♦7 ♥K ♣6 ♦9 ♣Q ♠6 ♣2 ♥7 ♥D ♦6 ♥9 ♣J ♠8 ♦D ♥J
Leonard: ♦4 ♣9 ♣D ♠2 ♠3 ♠4 ♣4 ♦3 ♣7 ♣K ♥5 ♥A ♥4 ♣5 ♦K ♠A ♦2 ♠9 ♠5 ♦J ♣3 ♠7 ♥8 ♥Q ♥6 ♣8
Venceu : S S S S E S L S S S S L S S L L S L S S S L S S
```

Na partida exemplificada acima, cada coluna corresponde a uma rodada. Na linha com o rótulo **Venceu**, um caractere 'S' indica que **Sheldon** venceu a rodada correspondente à coluna, um caractere 'L' indica que **Leonard** venceu a rodada correspondente a coluna, e um caractere 'E' indica que houve empate.

Como pode ser verificado, a primeira rodada foi vencida por **Sheldon** pois pegou o ♠K enquanto a carta que **Leonard** pegou foi o ♦4. Na quinta rodada, tivemos um empate, **Sheldon** e **Leonard** pegaram cartas de mesmo valor, o ♥3 e o ♠3. A primeira rodada vencida por **Leonard** foi a sétima quando pegou o ♣4 enquanto que **Sheldon** pegou o ♥2. O vencedor da partida foi **Sheldon** pois ele venceu 18 das 26 rodadas.

As constantes definidas abaixo devem ser **obrigatoriamente** utilizadas nessa questão, sem precisar redefini-las. O valor de cada carta depende apenas de seu posto: 'A', '2', '3', '4', '5', '6', '7', '8', '9', 'D', 'J', 'Q', e 'K'. O dicionário **VALOR** tem como **chave** o posto de cada carta e apresenta o seu **valor**.

```
# naipes, postos e valores de cada carta de acordo com o posto
NAIPES = ['♥', '♣', '♦', '♠']
POSTOS = ['A', '2', '3', '4', '5', '6', '7', '8', '9', 'D', 'J', 'Q', 'K']
VALOR = {'A':14, '2':2, '3':3, '4':4, '5':5, '6':6, '7':7, '8':8, '9':9,
         'D':10, 'J':11, 'Q':12, 'K':13}
```

<sup>1</sup>Esse jogo foi desenvolvido para muitas plataformas, incluindo Android, Apple IOS e Windows Phones. Como não há decisões no jogo e o resultado é totalmente aleatório, **War** não é considerado um jogo segundo algumas definições [Wikipedia].

### Item (a) (vale 1,0 ponto)

Escreva uma classe `Carta` que será usada para criar um objeto que representa uma carta de um baralho. Essa classe deve obedecer as especificações listadas a seguir.

- **Construtor da classe:** deve receber como parâmetros um string `naipe` (da lista `NAIPES`) e um string `posto` (da lista `POSTOS`) que serão usados para criar a representação de uma carta.
- `pegue_naipe()`: a classe deve possuir um **método** `pegue_naipe()` que **retorna** o naipe de um objeto da classe.
- `pegue_posto()`: a classe deve possuir um **método** `pegue_posto()` que **retorna** o posto de um objeto da classe.
- `pegue_valor()`: a classe deve possuir um **método** `pegue_valor()` que **retorna** o valor de um objeto da classe. Esse método deve utilizar o dicionários `VALOR`.

### Item (b) (vale 1,0 ponto)

Escreva uma classe **Baralho** que será usada para criar um objeto que representa um baralho. Essa classe deve obedecer as especificações listadas a seguir.

- **Construtor da classe:** para criar a representação de um baralho, o **construtor da classe** deve utilizar uma lista (= `list`) de objetos da classe **Carta** (item (a)). A representação de cada uma das 52 cartas de um baralho deve estar presente na lista.
- **embaralhe():** a classe deve possuir um **método embaralhe()** que **embaralha** a lista de cartas de um dado objeto da classe. Para isso utilize a função `random.shuffle()` que **recebe** uma lista e embaralha os seu elementos. Não escreva a função `random.shuffle()`, apenas use-a.
- **pegue\_carta():** a classe deve possuir um método `pegue_carta()` que **remove** e **retorna** a carta no final da lista de cartas de um dado objeto da classe.





### Item (c) (vale 1,0 ponto)

Escreva uma classe **Jogador** que será usada para criar um objeto que representa um jogador de **War**. Essa classe deve obedecer as especificações listadas a seguir.

- **Construtor da classe:** recebe como parâmetro um string **nome** com o nome do jogador. Um jogador deve ser representado através de um string com o **seu nome** e uma lista contendo a representação das **cartas no seu monte**. No momento da criação, o monte de cartas do jogador está vazio. Cada uma das cartas no monte do jogador deve ser representada através de objetos da classe **Carta** (item (a)).
- A classe deve possuir um **método** que permita que ao utilizarmos a função **print()** com um objeto da classe **Jogador**, o nome do jogador e as todas as cartas no seu monte sejam mostrados como nos exemplos abaixo:

Sheldon: ♡8 ♡7 ◇7 ♠J ♡5 ♣J ◇3 ♠Q ♣Q ♣K

Leonard: ◇A ♣4 ♣2 ♡2 ♠2 ♡D ♣9 ♡A ♣6 ◇9 ♣7 ♡4 ◇4 ◇5 ♠K ♠3 ♠D ♡6

- **recebe\_carta():** a classe deve possuir um **método** **recebe\_carta()** que **recebe** como parâmetro um objeto **carta** da classe **Carta** (item (a)) e **põe** a carta no topo do monte de cartas de um dado objeto da classe.
- **pegue\_carta():** a classe deve possuir um método **pegue\_carta()** que **remove** e **retorna** a carta no topo do monte de cartas de um dado objeto da classe.



### Item (d) (vale 2,0 pontos)

Neste item você deverá escrever uma função `main()` que simula uma partida de War entre os jogadores Sheldon e Leonard. A sua função deverá utilizar objetos das classes `Carta` (item (a)), `Baralho` (item (b)) e `Jogador` (item (c)). A seguir estão três exemplos de execução do programa.

War entre Sheldon e Leonard

```
Sheldon: ♣9 ♦9 ♣4 ♣5 ♦6 ♦Q ♣K ♠4 ♥6 ♣6 ♣2 ♠D ♣J ♠K ♠Q ♥A ♦8 ♥7 ♦A ♣D ♣3 ♦5 ♣7 ♠6 ♦7 ♦J
Leonard: ♦4 ♦D ♥3 ♦K ♠3 ♠2 ♠8 ♠7 ♥D ♦2 ♥8 ♥9 ♥2 ♥5 ♥J ♣Q ♣A ♣8 ♠J ♥Q ♦3 ♠9 ♠A ♥4 ♠5 ♥K
Venceu : S L S L S S S L L S L S S S S S L L S L E L L S S L
Sheldon venceu a partida.
```

War entre Sheldon e Leonard

```
Sheldon: ♦7 ♥3 ♠9 ♥2 ♠J ♦8 ♦4 ♣J ♣4 ♠7 ♣8 ♥A ♣A ♦A ♠5 ♣K ♦Q ♦2 ♥Q ♥9 ♥D ♠4 ♠3 ♣5 ♠K ♣Q
Leonard: ♥7 ♠Q ♣9 ♣7 ♥K ♥8 ♣6 ♦9 ♦J ♥6 ♣2 ♥4 ♠2 ♠A ♦D ♠8 ♦6 ♣3 ♦K ♥J ♠6 ♠D ♦5 ♦3 ♥5 ♣D
Venceu : E L E L L E L S L S S S S E L S S L L L S L L S S S
Sheldon e Leonard empataram a partida.
```

War entre Sheldon e Leonard

```
Sheldon: ♥3 ♠2 ♥K ♦Q ♥9 ♠3 ♦2 ♣7 ♣A ♣3 ♠5 ♣D ♥7 ♦9 ♥8 ♠D ♠J ♥6 ♣2 ♦7 ♥A ♥D ♠K ♥2 ♠8 ♣6
Leonard: ♦8 ♠7 ♠4 ♦3 ♥J ♥Q ♣9 ♣J ♠Q ♦K ♦4 ♣5 ♠9 ♦J ♥4 ♠6 ♣K ♣4 ♣Q ♦6 ♥5 ♦A ♣8 ♦D ♦5 ♠A
Venceu : L L S S L L L L S L S S L L S S L S S L S L S L
Leonard venceu a partida.
```

As mensagens emitidas pelo seu programa devem ser idênticas às mensagens mostradas no exemplo. Na simulação de uma partida um baralho deve ser **criado** e **embaralhado**. Cada um dos jogadores deve **receber** metade das 52 cartas do baralho. Cada carta **recebida** por um jogador é colocada no seu **monte**. Linhas com os nomes dos jogadores seguidos pelas cartas nos seus montes devem ser **impressas**, como mostra o exemplo. Em cada uma das rodadas, as cartas no topo dos montes dos jogadores são comparadas. No final da partida **devem impressas duas linhas: uma linha** contendo o rótulo **Venceu:** seguido das marcas correspondentes aos vencedor de cada rodada (= **SHELDON** ou **LEONARD**) ou ainda a marca representando empate (= **EMPATE**) e **uma linha** indicando o jogador que venceu a partida ou se houve empate (veja os exemplos de execução).

A seguir estão algumas constantes que podem ser usadas pela sua função `main()`.

```
# resultados de uma rodada
```

```
SHELDON = 'S'
```

```
LEONARD = 'L'
```

```
EMPATE = 'E'
```

```
# número de cartas e rodadas
```

```
N_CARTAS = 52
```

```
N_RODADAS = 26
```

