1 Exercícios Dicionários

1. **Adultos**: Crie um dicionário com nome, idade e telefone de 10 pessoas (os dados serão digitados pelo usuário). Depois, remova todos os usuários que forem menores de idade, e imprima nome/telefone dos que permanecerem.

2. **Agenda**: Faça um programa no qual o usuário possa armazenar o número de telefone de quantos contatos ele quiser. O programa tem um menu com duas opções:

   - Cadastrar telefone: Usuário digita nome e telefone de um novo contato.
   - Visualizar agenda: Programa exibe nomes e telefones cadastrados.

   As informações cadastradas precisam ser salvas em um arquivo texto. Enquanto estiverem na memória, serão armazenadas em um *dicionário*.

3. **Bolão - Dicionários**: Adapte o código do Bolão para que os dados dos jogadores sejam salvos em um dicionário, ao invés de uma lista. Garanta, ainda, que todos os dados cadastrados serão salvos em um arquivo binário e lidos automaticamente no início do programa.

4. **Concurso Público**: Um certo concurso público oferece vagas em diferentes áreas. Cada candidato possui um número de inscrição, nome, data de nascimento, nota na primeira fase e nota na segunda fase. Cada área possui um código, um nome e uma lista com o número de inscrição de todos os candidatos inscritos nesta área. Crie funções para:

   (a) Dado um dicionário com os candidatos, o dicionário com as áreas, e o código de uma área específica, liste todos os candidatos classificados para a segunda etapa naquela área (apenas os candidatos com nota menor que 60 são eliminados na primeira etapa).

   (b) Dados o dicionário com os candidatos e o dicionário com as áreas, liste o candidato aprovado em cada área. A nota final é a soma das duas etapas e há apenas uma vaga para cada área. Em caso de dois candidatos com a mesma nota ficarem em primeiro lugar, é aprovado o candidato mais velho.

5. **Multas**: Enzo é um agente de trânsito que está cursando Sistemas de Informação e decidiu deixar o seu dia a dia mais informatizado. Ao invés de registrar as infrações de trânsito no papel, ele quer criar um aplicativo para utilizar em seu tablet. Ele decidiu salvar as informações com as seguintes estruturas:

   - Dicionário com dados dos **motoristas**: A chave do dicionário é o número da **CNH**, e o conteúdo é uma tupla com o **nome** do motorista e a data de vencimento da sua CNH.

- Dicionário com dados dos **veículos**: A chave é a **placa** do veículo, e o conteúdo é uma tupla contendo o número da **CNH** do proprietário, o **modelo** do veículo e a **cor** do veículo.
- Lista com dados das **infrações**: Cada elemento é uma tupla contendo o **código** da infração, a **data** (tupla com dia, mês e ano), **placa** do veículo multado e a **natureza** da infração.
- Dicionário com a **natureza** das infrações: A chave é o **tipo** da infração, e o conteúdo é a **pontuação** que o condutor recebe por ela.

Exemplo:

```
1  motoristas = { "01234567" : ("Seu Madruga",   (15,10,2019)),
2                 "12345678" : ("Dona Florinda", (14,10,2019))}
3
4  veiculos = { "FLA 1981": ("12345678", "Fusca",    "Preto"),
5               "ALE 2014": ("12345678", "Brasilia", "Prata"),
6               "BRU 0071": ("01234567", "Chevette", "Branco")}
7
8  infracoes = [("I0001",(15,10,2018),"BRU 0071","Gravissima"),
9               ("I0002",(16,10,2018),"BRU 0071","Gravissima"),
10              ("I0003",(17,10,2018),"ALE 2014","Leve") ]
11
12 naturezas = { "Leve" : 3, "Media" : 4,
13               "Grave" : 5, "Gravissima" : 7 }
```

Enzo já começou a desenvolver o sistema, mas está precisando de ajuda com algumas funcionalidades.

(a) Enzo quer filtrar as infrações ocorridas há menos de 1 ano, para que as demais deixem de contar pontos nas CNHs. Para isso, crie uma função que receba como parâmetros a lista de infrações e uma tupla contendo a data atual, e retorne uma nova lista contendo apenas as infrações que ocorreram há menos de 1 ano. Sugestão: Crie uma função auxiliar que receba duas datas, e verifique se a primeira ocorreu antes da segunda.

(b) Enzo quer calcular e retornar os pontos da CNH de um motorista. Para isso, crie uma função que receba como parâmetros a CNH do motorista, a lista de infrações e os dicionários que achar necessários, e retorne a pontuação atual do motorista. Observação: Considere que as infrações ocorridas há mais de 1 ano já foram removidas.

(c) Enzo quer consultar informações sobre um veículo e seu motorista em uma blitz. Para isso, crie uma função que receba como parâmetros a CNH do motorista, a placa do veículo, a data atual, a lista de infrações e os dicionários

que achar necessários. A função deve imprimir um alerta para o agente de transito apreender a CNH do motorista na seguintes situações: (1) Placa não cadastrada; (2) Motorista com CNH vencida; (3) Motorista com 20 pontos ou mais na CNH. Se as documentações do carro e do motorista estiverem em dia, imprimir o modelo e a cor do veículo cadastrado com aquela placa, nome do proprietário, pontuação do proprietário, nome do motorista e pontuação do motorista. Observação: Considere que as infrações ocorridas há mais de 1 ano já foram removidas, e utilize funções já implementadas nas questões (a) e (b) quando necessárias.

6. **Academia**: Enzo não estava fazendo sucesso como estudante de Sistemas de Informação e se matriculou em uma academia de musculação. Entretanto, como bom aluno, ficou inconformado ao perceber que sua academia ainda armazenava os treinos em fichas de papel. Enzo, portanto, resolveu desenvolver um aplicativo de celular para vender para sua academia. O aplicativo vai se chamar "Hoje Tá Pago"e deve armazenar as seguintes informações:

   (a) Um dicionário contendo todos os possíveis exercícios da academia. A chave é código do exercício e o valor é o nome do exercício. Exemplo:

```
exercicios =  { "P001" : "Supino articulado inclinado",
   "P002" : "Supino declinado com halteres",
   "P003" : "Fly", "C001" : "Barra graviton",
   "C002" : "Fly inverso",
   "C003" : "Remada curvada supinada",
   "T001" : "Triceps maquina",
   "T002" : "Triceps testa com barra",
   "T003" : "Triceps inverso",
   "B001" : "Biceps barra W",
   "B002" : "Biceps alternado polia baixa",
   "B003" : "Rosca neutra com halteres",
   "A001" : "Abdominal obliquo banco lombar",
   "A002" : "Abdominal paralela joelho estendido",
   "A003" : "Abdominal canivete",
   "A004" : "Abdominal morcego",
   "A005" : "Lombar banco",
   "O001" : "Desenvolvimento Arnold",
   "O002" : "Elevacao frontal com corda",
   "O003" : "Remada alta na polia baixa",
   "I001" : "Agachamento Smith",
   "I002" : "Leg horizontal",
   "I003" : "Cadeira extensora",
   "I004" : "Flexora vertical",
```

```
24     "I005" : "Cadeira flexora",
25     "I006" : "Panturrilha maquina" }
```

(b) Um dicionário para armazenar os dados dos alunos. A chave é o login do aluno, e o conteúdo é uma tupla contendo seu nome e senha. Exemplo:

```
1 alunos = { "enzo" : ("Enzo da Silva", "12345"),
2            "vale" : ("Valentina", "abacaxi"),
3            "ramon" : ("Seu Madruga", "b71<3") }
```

(c) Um dicionário com todos os treinos cadastrados no aplicativo. A chave do dicionário é o login do aluno (portanto, só há um treino para cada aluno). O conteúdo de cada elemento do dicionário é uma lista de atividades que o aluno deve fazer, e cada atividade é representada por uma tupla contendo 4 informações:

- Código do exercício
- Quantidade de séries que o aluno deve executar
- Número de repetições em cada série
- Grupo do treino (o aluno pode fazer grupos diferentes de exercícios por dia).

Exemplo:

```
1  treinos = { "enzo" : [("P001",3,8,"A"),
2                  ("P002",3,8,"A"), ("P003",3,8,"A"),
3                  ("T001",3,8,"A"), ("T002",3,8,"A"),
4                  ("T003",3,8,"A"), ("A001",4,15,"A"),
5                  ("A002",4,15,"A"), ("C001",3,8,"B"),
6                  ("C002",3,8,"B"), ("C003",3,8,"B"),
7                  ("B001",3,8,"B"), ("B002",3,8,"B"),
8                  ("B003",3,8,"B"), ("A001",3,8,"B"),
9                  ("A002",4,15,"B"), ("A003",4,15,"B"),
10                 ("O001",4,15,"C"), ("O002",3,8,"C"),
11                 ("O003",3,8,"C"), ("P001",3,8,"C"),
12                 ("P002",3,8,"C"), ("P003",3,8,"C"),
13                 ("P004",3,8,"C"), ("P005",3,8,"C"),
14                 ("P006",3,8,"C") ],
15             "vale" : [("P001",3,8,"A"),
16                 ("T001",3,8,"A"), ("A001",3,8,"A"),
17                 ("C001",3,8,"B"), ("B001",3,8,"B"),
18                 ("A001",3,8,"B"), ("A003",3,8,"C"),
19                 ("O001",3,8,"C"), ("P006",3,8,"C") ] }
```

Enzo já começou a desenvolver o sistema, mas está precisando de ajuda com algumas funcionalidades.

- Crie uma função que receba como parâmetros os três dicionários, o login de um aluno e um grupo. A função deve imprimir o nome do aluno e quais atividades daquele grupo que o aluno deverá fazer. Para login="enzo" e grupo="A", por exemplo, a saída seria:

```
Aluno: Enzo
Grupo: A

Supino articulado inclinado - 3 de 8
Supino declinado com halteres - 3 de 8
Fly - 3 de 8
Triceps maquina - 3 de 8
Triceps testa com barra - 3 de 8
Triceps inverso - 3 de 8
Abdominal obliquo banco lombar - 4 de 15
Abdominal paralela joelho estendido - 4 de 15
```

- Crie uma função que receba como parâmetros os três dicionários e o login de um aluno, e imprima o nome de todos os exercícios que o aluno ainda não faz.
- Crie uma função para autenticar o aluno no sistema. A função deve solicitar que o usuário digite seu login e senha, e então validar se a senha está correta. Caso esteja, a função solicita que o aluno informe o grupo que ele deseja treinar naquele dia, e então utilizar a função da Questão 1 para exibir os exercícios daquele grupo que o aluno deve realizar. A função deve exibir mensagens de erro diferentes para os seguintes casos: login não existente, senha incorreta, ou grupo inexistente para o aluno (ou seja, se nenhum exercício tiver sido cadastrado para o aluno naquele grupo).

7. **Produtos**: Uma certa loja possui um programa que mantém armazenados os produtos vendidos por ela, os clientes cadastrados e o histórico de pedidos já realizados, ou seja, dos produtos que já foram vendidos. Para isso, o sistema armazena 3 dicionários na memória:

   - O dicionário **clientes**, contendo cpf (String), nome (String) e endereço eletrônico (String) de cada cliente. A chave do dicionário é o cpf.
   - O dicionário **produtos**, contendo o código do produto (String), o valor unitário (double), o nome do produto (String) e a quantidade em estoque (inteiro). A chave do dicionário é o código do produto.

- O dicionário **pedidos**, contendo o código do pedido (String), o código do cliente que realizou o pedido, o status do pedido (verdadeiro caso ele já tenha sido entregue, ou falso caso ainda esteja em aberto) e a lista de produtos que o cliente comprou. Cada elemento da lista de produtos é uma tupla, contendo o código de um produto e a quantidade de itens daquele produto que o cliente solicitou. A chave do dicionário é o código do pedido.

Exemplo:

```
clientes = {"123.456.789-00" :
                    ("Maria da Silva", "maria@gmail.com"),
            "234.567.890-00" :
                    ("Joao da Cruz", "joao@gmail.com"),
            "345.678.900-00" :
                    ("Jose de Souza", "zezinho@gmail.com") }

produtos = {"A0001" : (1.20, "Pera", 1),
            "A0002" : (3.40, "Uva", 1),
            "A0003" : (1.00, "Maca", 1),
            "A0004" : (10.00, "Salada de frutas", 1),
            "A0005" : (12.00, "Acai medio", 1),
            "A0006" : (3.00, "Granola", 1),
            "A0007" : (5.00, "Suco 300ml", 1) }

pedidos = {"345" : ("123.456.789-00", True,
                    [("A0001",3), ("A0002",1),
                    ("A0003",5), ("A0004",1)]),
            "123" : ("234.567.890-00", False,
                    [("A0005",2), ("A0006",1)]) }
```

Crie as seguintes funções:

(a) Crie uma função que receba como parâmetro o código de um produto e os três dicionários, e verifique se há quantidade em estoque suficiente para todos os pedidos em aberto que envolvem aquele produto.

(b) Crie uma função que receba como parâmetro o código de um pedido e os três dicionários, e imprima o pedido no seguinte formato:

Exemplo 1:

```
Pedido #345:
- Pera
  Qtd: 3
  Valor unitário: R$ 1,20
```

```
  Valor total: R$ 3,60
- Uva
  Qtd: 1
  Valor unitário: R$ 3,40
  Valor total: R$ 3,40
- Maçã
  Qtd: 5
  Valor unitário: R$ 1,00
  Valor total: R$ 5,00
- Salada de frutas
  Qtd: 1
  Valor unitário: R$10,00
  Valor total: R$ 10,00
Status: Entregue
```

Exemplo 2:

```
Pedido #123:
- Açaí médio
  Qtd: 2
  Valor unitário: R$ 12,00
  Valor total: R$ 24,00
- Granola
  Qtd: 1
  Valor unitário: R$ 3,00
  Valor total: R$ 3,00
Status: Em aberto
```

(c) Crie uma função que receba como parâmetro o código de um pedido e os três dicionários, e retorne o valor total daquele pedido.

(d) Crie uma função que receba como parâmetro os três dicionários, e imprima o nome de cada cliente ao lado do valor total já pago por ele na loja. Para calcular o valor de cada pedido feito por um cliente, utilize a função da Questão (c).

8. **Voos**: Uma agência de turismo possui armazenados os voos realizados por diversas companhias aéreas. Esses voos são representados como um dicionário que contém as informações pertinentes a cada um desses voos. A chave do dicionário é um número inteiro com o número do voo, e o conteúdo é uma tupla contendo:

   - Companhia que realizou o vôo (String).
   - Data do voo (tupla contendo inteiros positivos para o dia, mês e ano).

- Lista de escalas. Cada elemento da lista é uma tupla contendo uma cidade (String) e um horário (tupla contendo a hora e o minuto, ambos inteiros). O horário na cidade origem corresponde ao horário de saída do voo. A última tupla da lista corresponde sempre ao destino final. O horário nas tuplas seguintes corresponde à chegada em cada uma das demais escalas. Considere que o tempo de voo do trecho entre duas escalas é a diferença (em minutos) dos horários relativos às duas escalas.

Exemplo:

```
voos = {1024 : ("TAM",  (11, 9, 2001), [("ES", (11, 30)),
                                         ("RJ", (12, 30)),
                                         ("SP", (13, 50)),
                                         ("NY", (22, 00)) ]),
        1025 : ("GOL", (11, 9, 2001), [("ES", (14, 00)),
                                       ("SP", (16, 00)) ]) }
```

Implemente as funções a seguir:

(a) Dados o número de um voo x e o dicionário com os voos, calcule e retorne o tempo total (em minutos) daquele voo. (Cuidado: um voo pode começar em um dia e terminar no outro).

(b) Dados os voos, exiba o número de cada voo seguido de suas cidades de origem e de destino. Em cada linha do resultado deverá aparecer o número de um voo e o nome das duas cidades.

(c) Dados uma cidade origem a, uma cidade destino b e o dicionário com os voos, imprima o número de todos os voos que se iniciem em a e passem por b.

(d) Dados uma cidade origem a, uma cidade destino b e o dicionário com os voos, imprima a companhia e o número do voo de menor tempo total que saia de a e termine em b. Utilize a função do item 8a para calcular o tempo total de um voo.

9. **Séries**: Empolgado com a disciplina de programação, você decidiu implementar um sistema para gerenciar as séries de TV que você assiste no pouquíssimo tempo livre que possui. O sistema permite que você:

- Cadastre as séries que assiste
- Delete uma série (caso tenha cadastrado errado ou desistido de acompanhá-la)
- Insira a quantidade de episódios que cada temporada da série possui
- Controle os episódios que você já assistiu em cada série

Por enquanto, a interface do seu sistema será feita pelo terminal, e exibirá as seguintes opções:

(a) Cadastrar série: Solicita que o usuário informe o nome da série a ser cadastrada. A série digitada pelo usuário só é inserida caso ainda não conste no sistema.

(b) Excluir série: Exibe o nome das séries já cadastradas, e pede que o usuário informe aquela que deseja excluir.

(c) Cadastrar nova temporada: Permite que seja cadastrada uma nova temporada de uma série. Para isso, o sistema exibe o nome das séries cadastradas e pede que o usuário informe aquela em que deseja adicionar uma nova temporada. Só é possível cadastrar uma nova temporada por vez, e elas são adicionadas em ordem. A temporada 3, por exemplo, só pode ser adicionada depois que as temporadas 1 e 2 forem previamente adicionadas. Após o usuário escolher o nome da série, o sistema informa quantas temporadas já foram cadastradas, e pergunta ao usuário quantos episódios existem na nova temporada.

(d) Exibir séries: Lista o nome de todas as séries já cadastradas, além de exibir os episódios de cada temporada. O episódio 8 da temporada 2 de uma série, por exemplo, é identificado como T2E8 (Temporada 2, Episódio 8). Além disso, o sistema usa o símbolo '-' para identificar um episódio que você ainda não assistiu, e o símbolo '+' para identificar um episódio já assistido. Exemplo:

```
* Westworld *

 Temporada 1: +T1E1 +T1E2 +T1E3 +T1E4 +T1E5 -T1E6 -T1E7 -T1E8
 Temporada 2: -T2E1 -T2E2 -T2E3 -T2E4 -T2E5 -T2E6 -T2E7 -T2E8

* Game of Thrones *

 Nenhuma temporada cadastrada
```

Note que, no exemplo acima, apenas os cinco primeiros episódios de "Westworld" foram marcados como assistidos até agora.

(e) Marcar episódio como visto: Exibe todas as informações da opção para ajudar o usuário a identificar o episódio que deseja marcar como visto. Em seguida, o sistema solicita 3 informações:

i. O nome da série
ii. O número da temporada em que o episódio assistido se encontra
iii. O número do episódio que deve ser marcado como assistido.

(f) Marcar temporada como vista: Exige que o usuário informe apenas o nome

da série e qual temporada será marcada como assistida. Em seguida, todos os episódios da temporada escolhida serão marcados como assistidos no sistema.

(g) Marcar série como vista: Solicita apenas que o usuário informe o nome de uma série, e marca todos os episódios de todas as temporadas desta série como assistidos.

(h) Salvar estatísticas: Cria um arquivo texto chamado "estatisticas.txt" (na mesma pasta em que o programa está salvo) informando:

- A quantidade de séries cadastradas
- O nome das séries que você está em dia (ou seja, já assistiu todos os episódios possíveis)
- O nome das sérias atrasadas, com os respectivos episódios que ainda não foram assistidos
- A quantidade total de episódios cadastrados
- A quantidade total de episódios que você assistiu.

A seguir, um exemplo possível para o arquivo de estatísticas:

```
Quantidade de series cadastradas: 8

Series em dia:
* Dexter
* Spartacus
* The Mentalist
* Game of Thrones
* Breaking Bad

Episodios atrasados:
* Sense8: T2E1 T2E2 T2E3 T2E4 T2E5 T2E6 T2E7 T2E8 T2E9 T2E10 T2E11
* Gotham: T4E8 T4E9 T4E10 T4E11 T4E12 T4E13 T4E14 T4E15 T4E16 T4E17 T4E18 T4E19 T4E20 T4E21 T4E22
* Westworld: T1E1 T1E2 T1E3 T1E4 T1E5 T1E6 T1E7 T1E8 T1E9 T1E10

Quantidade total de episodios cadastrados: 536
Quantidade total de episodios ja assistidos: 500
```

Figura 1: Estatísticas.

As informações sobre as séries serão armazenadas na memória através de um dicionário. Cada chave do dicionário é o nome de uma série (não haverá mais de uma série com o mesmo nome). O valor de cada chave contém os episódios de cada temporada que foi cadastrada na série, informando ainda se cada episódio foi visto ou não. Por exemplo, suponha que a série "Sense8 " possui apenas duas temporadas cadastradas (a primeira com 12 episódios, e a segunda com 11), e que apenas os episódios da primeira temporada foram assistidos. Além disso, suponha que a série "Westworld" foi inserida no sistema, mas que nenhuma temporada foi cadastrada ainda. Neste caso, o dicionário irá armazenar as seguintes informações:

```
1 { 'Sense8': [[True, True, True, True, True, True, True,
2               True, True, True, True, True],
```

```
3               [False, False, False, False, False, False,
4                False, False, False, False, False]],
5   'Westworld': []
6 }
```

Como podemos notar no exemplo acima, o valor de cada chave no dicionário é uma lista. Cada elemento da lista corresponde a uma temporada e armazena uma sublista contendo os episódios cadastrados (em ordem). Episódios marcados com False ainda não foram assistidos, e episódios marcados com True já foram assistidos. A princípio, todo episódio é inserido como "não assistido".

Todas as informações cadastradas no sistema devem ser salvas em disco, num arquivo binário. Sempre que o sistema for iniciado, devem ser carregadas automaticamente as informações sobre as séries que foram previamente cadastradas.

10. **Perícia Criminal**: Você se formou no Bacharelado em Sistemas de Informação e foi aprovado no concurso para Perito Criminal da Polícia Federal. Sua primeira investigação envolve uma fraude bancária, e você precisa identificar inconsistências no Banco de Dados de um banco. As informações que você recebeu estão armazenadas nas seguintes estruturas:

    - **Clientes**: Dicionário com o **cpf** (chave) e o **nome** de cada cliente. Exemplo:

```
clientes = {'814.457.764-72': 'Albino Caballero',
            '063.521.442-33': 'Adao Pontes',
            '401.424.725-22': 'Aarao Rodovalho',
            '455.073.756-58': 'Abilio Mascarenas',
            '350.870.754-79': 'Adosindo Valladares'}
```

    - **Contas**: Dicionário com o **número da conta** (chave), **saldo** atual e **cpf** do cliente titular da conta. Obs.: Cada cliente pode possuir uma, nenhuma ou várias contas. Exemplo:

```
contas = {'006': (46, '401.424.725-22'),
          '005': (49, '401.424.725-22'),
          '001': (-22528, '401.424.725-22'),
          '008': (-81283, '063.521.442-33'),
          '007': (-71714, '401.424.725-22'),
          '004': (-98250, '455.073.756-58'),
          '002': (105, '455.073.756-58'),
          '003': (0, '814.457.764-72')}
```

    - **Histórico**: Dicionário com o histórico de transações a serem investigadas. A chave é o **identificador da transação**, e o conteúdo é o **valor** da transação,

o **número da conta** envolvida na transação e o **tipo de operação** (0 para saques e 1 para depósitos). Exemplo:

```
historico = {'00020': (14, '006', '1'),
             '00005': (17, '002', '1'),
             '00011': (23, '007', '1'),
             '00018': (15, '005', '1'),
             '00013': (29, '008', '1'),
             '00001': (53, '007', '1'),
             '00014': (35, '002', '1'),
             '00006': (13, '001', '0'),
             '00007': (67, '007', '0'),
             '00008': (67, '008', '0'),
             '00017': (40, '007', '1'),
             '00015': (20, '002', '1'),
             '00019': (95, '001', '1'),
             '00010': (15, '007', '1'),
             '00012': (6, '007', '1'),
             '00003': (65, '002', '1'),
             '00002': (32, '006', '1'),
             '00009': (32, '002', '0'),
             '00004': (9, '007', '1'),
             '00016': (34, '005', '1')}
```

O saldo de toda conta deveria refletir seu histórico de movimentações. Note que o histórico do exemplo indica apenas dois depósitos na conta '006' (um de R$14 e outro de R$32). Portanto, seu saldo de R$46 no dicionário de contas está correto. Entretanto, o saldo de R$-22.528 na conta '001' não reflete suas movimentações no histórico. Isso ocorre porque algumas contas foram invadidas por hackers, que retiraram dinheiro das contas sem deixar rastros no histórico de transações (as contas possuíam um saldo zerado no momento em que foram criadas).

(a) Crie uma função que receba os dicionários com o histórico e com as contas, e retorne um novo dicionário contendo o número de cada conta (chave) e o saldo correto que ela deveria ter de acordo com o histórico.

(b) Crie uma função que receba o dicionário de contas e o dicionário com saldos corretos (criado na função da questão 1) e retorne um novo dicionário com as contas invadidas. Para cada conta hackeada, armazene o número da conta (chave), o saldo que ela possui após a invasão, e o saldo que deveria possuir de acordo com o histórico.

(c) Crie uma função que receba os dicionários de contas, de contas invadidas (criado na função da questão 2) e de clientes, e imprima o número da conta,

cpf e nome do titular de todas as contas hackeadas que ficaram com saldo negativo e podem estar pagando juros do cheque especial indevidamente por causa da invasão.

(d) Crie uma função que receba apenas o dicionário de contas invadidas, e imprima o valor total que o banco deverá gastar para ressarcir seus clientes, considerando todo o valor que os hackers retiraram das contas indevidamente.