

**ATENÇÃO:** A sua resposta a essa lista deve ser um arquivo zip contendo uma pasta para cada questão: Q1, Q2, Q3, .... Na pasta raiz coloque um arquivo LEIAME.TXT, com os nomes e matrículas dos membros da equipe que está entregando o trabalho. O conteúdo de cada pasta varia com a questão, conforme explicado em cada uma delas. Nas questões que requerem envio de código, faça comentários, pois será critério de avaliação.

- 1) Faça um programa que lê três parâmetros:
  - a) Nome de um arquivo origem;
  - b) Uma palavra-passe;
  - c) Nome de um arquivo destino.

Sobre cada um dos bytes do arquivo de origem aplica uma operação de 'xor', considerando o valor ASCII das letras da palavra-passe. Assim, se a palavra-passe for 'pato', o primeiro byte do arquivo destino é o 'xor' do respectivo primeiro byte do arquivo de origem com o código ASCII do 'p'; o segundo byte usa o código ASCII do 'a' para gerar o segundo byte do arquivo de destino, a partir do segundo do arquivo de origem. E assim sucessivamente. Após o uso do último byte da palavra-passe, volte a usar o primeiro e o processo se repete.

Não esqueça de tratar as exceções. Não sobrescreva arquivos existentes, notifique o usuário nesses casos. Você deve entregar somente o programa (com comentários).

- 2) A dificuldade de minerar bitcoins envolve ocorre porque é necessário executar o que se chama de prova de trabalho. Em outras palavras, vários mineradores competem para realizar uma tarefa; aquele que primeiro realizar é o minerador campeão da atividade e recebe uma boa recompensa. Na prática, a atividade a realizar é: receber um conjunto de transações (um conjunto de bytes) e calcular o *hash* SHA-256 deles, mas tem um detalhe: um número de quatro bytes deve ser adicionado no início dos *bytes* recebidos (chame-o de *nonce*) e dos 256 bits de resultado uma determinada quantidade inicial deve ser zero. O minerador que descobrir o *nonce* certo é o vencedor. Gráficamente:



Portanto, minerar é: a) escolher um *nonce*; b) juntar com os *bytes* da entrada; c) calcular o *hash* desse conjunto; d) verificar se o *hash* resultante inicia com uma certa quantidade de *bits* em zero; e) se o *hash* calculado não atende ao requisito, repetir o processo.

Faça uma função em Python de nome `findNonce` que recebe três argumentos:

- `dataToHash` – um conjunto de bytes

- bitsToBeZero – o número de bits iniciais que deve ser zero no *hash* e devolve:

- o nonce encontrado
- o tempo (em segundos) que demorou para encontrar o nonce

Ao final, faça um programa que usa a função para preencher a seguinte tabela:

Texto a validar (converta para bytes antes de chamar)	Bits em zero	Nonce	Tempo (em s)
“Esse é fácil”	8		
“Esse é fácil”	10		
“Esse é fácil”	15		
“Texto maior muda o tempo?”	8		
“Texto maior muda o tempo?”	10		
“Texto maior muda o tempo?”	15		
“É possível calcular esse?”	18		
“É possível calcular esse?”	19		
“É possível calcular esse?”	20		

Sua resposta deve ser dois arquivos: o programa (com a função e outras auxiliares, se necessário) e a **tabela preenchida** (em formato doc, PDF ou txt).

- 3) Faça o download do arquivo dados\_cartola\_fc.rar (em anexo) e descompacte-o. Ele contém dados do *fantasy game* denominado Cartola. Leia os arquivos e se familiarize com os dados. Pergunte ao professor, se necessário.

Desenvolva um programa que atenda aos seguintes requisitos:

- O programa deverá solicitar ao usuário o ano em que se deseja acessar os dados do Cartola FC;
- Uma vez informado o ano, o programa deverá abrir o arquivo correspondente. Lembre-se de tratar possíveis;
- Caso o arquivo tenha sido lido com sucesso, o deverá solicitar ao usuário um dos esquemas táticos conforme tabela a seguir:

Esquema	Quantidade de Jogadores:
3-4-3	3 zagueiros / 0 laterais / 4 meias / 3 atacantes
3-5-2	3 zagueiros / 0 laterais / 5 meias / 2 atacantes
4-3-3	2 zagueiros / 2 laterais / 3 meias / 3 atacantes
4-4-2	2 zagueiros / 2 laterais / 4 meias / 2 atacantes
4-5-1	2 zagueiros / 2 laterais / 5 meias / 1 atacantes
5-3-2	3 zagueiros / 2 laterais / 3 meias / 2 atacantes
5-4-1	3 zagueiros / 2 laterais / 4 meias / 1 atacantes

- Para cada esquema tático, deve-se selecionar a quantidade de jogadores por posição obedecendo a tabela do item c dessa questão;
- Independente do esquema tático selecionado, todos terão de ter 1 goleiro e 1 técnico;

- f) A escolha dos atletas de cada posição será através daqueles que tiverem a maior pontuação (média de pontos \* quantidade de partidas) em cada posição (zagueiro, lateral, meia, atacante, goleiro, técnico);
- g) O programa deverá exibir na tela a lista dos atletas selecionados, mostrando a sua posição (zagueiro, lateral, meia, atacante, goleiro, técnico), o seu nome abreviado, o seu time e a sua pontuação (média de pontos x quantidade de partidas);
- h) O programa deverá salvar um arquivo contendo os dados exibidos no item g dessa questão:
  - i. Adicionar também a URL da foto do jogador e a URL do escudo do time do jogador;
  - ii. O nome do arquivo deverá ser `selecao_cartola_fc_nnnn.txt`, onde `nnnn` é o ano informado;
  - iii. Os dados de cada jogador deverão ser separados por ; (ponto e vírgula);
  - iv. A primeira linha do arquivo deverá ser:  
`posição;nome;url_foto_atleta;pontuação;time;url_escudo_time`

Você deve entregar somente o programa (com comentários).

- 4) Nos anos 80, Van Jacobson, Steve McCanne e outros desenvolveram o `tcpdump` – uma ferramenta de captura de tráfego de rede. A própria ferramenta é capaz de decodificar o tráfego e apresentá-lo em maneira legível aos usuários. Mas também pode gravá-lo em formato binário, para leitura e análise posterior.

Para gravar o tráfego no `tcpdump` use o comando “`tcpdump -w nomeArquivo.cap`”. O formato do arquivo gravado é:

```
+-----+-----+-----+-----+-----+
| cabecalhoArquivo | pacote1 | pacote2 | pacote3 | ...
+-----+-----+-----+-----+-----+
```

O formato cabeçalho do arquivo é:

```

      1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Magic Number                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|      Major Version                   |      Minor Version                   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Reserved1                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Reserved2                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     SnapLen                                       |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| FCS |f|                               LinkType                                |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     |
|          Timestamp (Seconds)       |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     |
|          Timestamp (Microseconds or nanoseconds)       |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     |
|          Captured Packet Length    |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     |
|          Original Packet Length    |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
/                                     /
/          Packet Data                /
/          variable length            /
/                                     /
+-----+

```

Desenvolva um programa que leia um arquivo capturado pelo `tcpdump` (alguns exemplos seguem em anexo) e responda:

- ATENÇÃO:** Não é permitido usar bibliotecas não nativamente incorporadas ao Python.

5) Faça um programa em Python que pergunte ao usuário o nome de um diretório e, para cada um dos arquivos nele presentes identifique aqueles se são imagens JPEG com informação de EXIF (iniciam com os *bytes* FF D8 FF E1). Para cada arquivo JPEG responda (se o dado existe):

- Ao final, apresente todas as cidades em que fotos foram capturadas e quantas em cada uma delas.

As informações de metadados de uma imagem JPEG iniciam na posição 2 do arquivo. Ali há uma grande estrutura com vários dados, denominada `app1Data` (veja <https://www.media.mit.edu/pia/Research/deepview/exif.html> para a descrição completa). Na posição 18 de `app1Data` há 2 bytes que indicam quantos metadados essa imagem tem.

A partir da posição 20 de `app1Data` (ou na 22 contada a partir do início do arquivo) há efetivamente os metadados. Cada metadado tem o formato:

- 2 bytes – qual o metadado, na forma de um identificador (*id*). Você pode obter a lista dos significados dos *ids* em: <https://exiftool.org/TagNames/EXIF.html>. Atente, em particular, para: 0x0100 (largura da imagem); 0x0101 (altura da imagem); 0x010F (fabricante da câmera); 0x0110 (modelo da câmera); 0x0132 (Data em que a imagem foi modificada); 0x9003 (Data em que a imagem foi capturada); 0x8769 (metadados adicionais de EXIF: lista de metadados, com o número deles nos dois primeiros dados); 0x8825 (informações de GPS: lista de metadados, com o número deles nos dois primeiros dados);
- 2 bytes – o tipo do metadado. Valores possíveis são, entre outros: (1 – unsigned byte; 2 – string; 3 – unsigned short; 4 – unsigned long, ...);
- 4 bytes – o número de repetições que esse metadado tem. Exemplo: tem tipo inteiro, mas se repete 5 vezes.
- 4 bytes – o valor do metadado. Se são necessários mais de 4 bytes, indica o *offset* no arquivo onde o valor está, contado a partir da posição 12 do início do arquivo (ou seja, deve somar 12 para chegar na posição real no arquivo).

As informações detalhadas sobre localização (latitude e longitude, por exemplo) presentes em uma imagem, podem ser obtidas aqui: <http://web.mit.edu/graphics/src/Image-ExifTool-6.99/html/TagNames/GPS.html>. Ressalte-se que a partir da latitude e da longitude, é possível obter os dados reais da localidade (na forma de um dicionário), tais como o endereço e CEP, usando os seguintes comandos:

```
import requests, json
local = json.loads(requests.get(
    "https://nominatim.openstreetmap.org/" +
    "reverse?" +
    "lat=-5.81085&lon=-35.20429" +
    "&format=json"
).text)
```

**ATENÇÃO:** Não é permitido usar bibliotecas não nativamente incorporadas ao Python, exceto requests.

Você deve entregar somente o programa (com comentários).