Lista de Exercícios #01: Revisão

Observações:

1. Os programas deverão ser desenvolvidos em linguagem PYTHON;
2. As funções deverão ser criadas em um arquivo em separado a fim de que sejam compartilhadas pelo programas a serem desenvolvidos nessa lista;
3. Agradecimentos ao Professor Galileu Batista de Sousa pela cessão do texto e arquivos relativos a questão 3;
4. Nas questões 3, 4 e 5 não está explícita a criação de funções nem os tratamentos de exceções, porém a utilização de funções criadas pelo aluno e o uso de exceções é critério avaliativo, ou seja, o não uso implica em decréscimo na nota de cada uma.

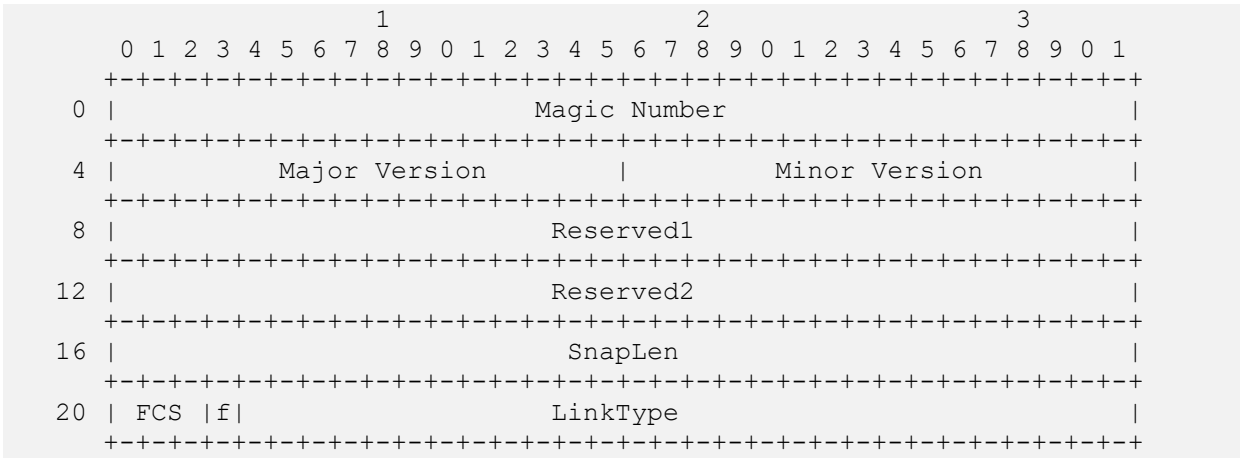
1. Elaborar um programa que siga as seguintes especificações:
   a) O programa deverá solicitar 3 valores inteiros sendo que, deverão ser tratadas as exceções caso os valores informados não sejam inteiros válidos;
   b) Uma vez informados os valores válidos, deverá ser chamada uma função (para fins de padronização iremos nomear essa função como ***gerar_lista***). Essa função terá as seguintes especificações:
      i. Ela irá receber 3 argumentos: ***quantidade***, ***valor_minimo*** e ***valor_maximo***;
      ii. Ela irá retornar ao programa 2 valores, o primeiro um valor booleano (***True*** ou ***False***) informando se a lista foi gerada corretamente e o segundo valor será a lista gerada propriamente dita (caso tenha sido gerada), caso contrário o segundo valor deverá ser ***None***;
      iii. A lista a ser gerada deverá conter a quantidade de valores informados no argumento ***quantidade***, Esses valores deverão estar compreendidos entre o argumento ***valor_minimo*** e ***valor_maximo***, não deverá ser ordenada e poderá conter repetições.
   c) Uma vez que a lista tenha sido gerada, deverá ser chamada uma função (para fins de padronização iremos nomear essa função como ***salvar_lista***). Essa função terá as seguintes especificações:
      i. Ela irá receber 2 argumentos: ***nome_lista***, ***nome_arquivo***;
      ii. Ela irá retornar ao programa 1 valor booleano (***True*** ou ***False***) informando se a lista foi salva corretamente ou não.
      iii. A lista informada no argumento ***nome_lista*** deverá ser salva no mesmo diretório/pasta da aplicação em um arquivo com nome informado no argumento ***nome_arquivo***;
      iv. Cada linha do arquivo deverá conter apenas um valor da lista informada.

2. Elaborar um programa que siga as seguintes especificações:
   a) O programa deverá solicitar o nome do arquivo salvo na questão 1 dessa lista de exercícios.
      i. Deverão ser tratadas as exceções que eventualmente possam surgir (atentem para caso o arquivo não exista).
   b) Uma vez que o nome do arquivo seja válido, deverá ser chamada uma função (para fins de padronização iremos nomear essa função como ***ler_arquivo***). Essa função terá as seguintes especificações:
      i. Ela irá receber 1 argumento: ***nome_arquivo***;
      ii. Ela irá retornar ao programa 2 valores. O primeiro será um valor booleano (***True*** ou ***False***) informando se o arquivo foi lido com sucesso ou não e o segundo valor será a lista gerada com base no arquivo lido. Caso o primeiro valor de retorno seja ***False***, o segundo valor de retorno deverá ser ***None***.
   c) Uma vez que já temos a lista lida do arquivo, deverá ser chamada uma função (para fins de padronização iremos nomear essa função como ***ordena_lista***). Essa função terá as seguintes especificações:
      i. Ela irá receber 2 argumentos: ***nome_lista*** e ***método_ordena***;
      ii. Ela irá retornar ao programa 2 valores. O primeiro será um valor booleano (***True*** ou ***False***) informando se a lista foi ordenada ou não e o segundo valor será a lista ordenada com base no método de ordenação informado no segundo argumento. Caso o primeiro valor de retorno seja ***False***, o segundo valor de retorno deverá ser ***None***;
      iii. O segundo argumento irá receber uma string informando o método de ordenação. As strings válidas para os métodos de ordenação são: ***BUBBLE***, ***INSERTION***, ***SELECTION*** e ***QUICK***;
      iv. Para cada um dos métodos de ordenação deverá ser criada uma função correspondente (para fins de padronização, cada função será nomeada da seguinte maneira: ***ordena_bubble***, ***ordena_insertion***, ***ordena_selection***, ***ordena_quick***) que deverá implementar o método de ordenação;
      v. Cada uma das funções de ordenação irá retornar ao programa 2 valores. O primeiro será um valor booleano (***True*** ou ***False***) informando se foi ordenada ou não e o segundo valor será a lista ordenada. Caso o primeiro valor de retorno seja ***False***, o segundo valor de retorno deverá ser ***None***.

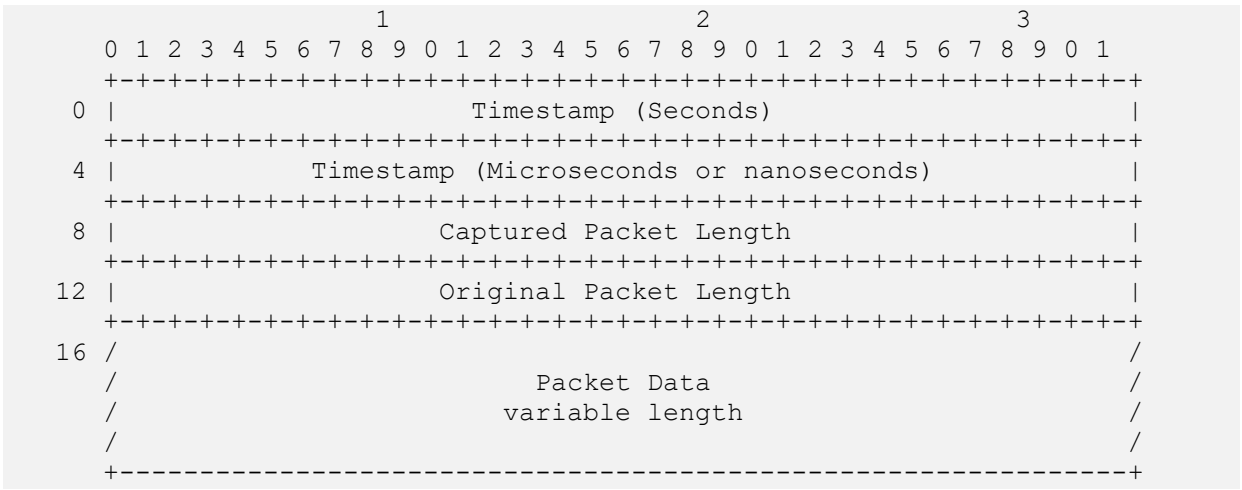
3. Nos anos 80, Van Jacobson, Steve McCanne e outros desenvolveram o tcpdump – uma ferramenta de captura de tráfego de rede. A própria ferrramenta é capaz de decodificar o tráfego e apresentá-lo em em maneira legível aos usuários. Mas também pode gravá-lo em formato binário, para leitura e análise posterior.

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

   O formado cabeçalho do arquivo é:



   E o formato de cada pacote que segue o cabeçalho do arquivo é:



Explicações para o significado de cada um dos campos nas figuras anteriores, bem como informações adicionais, podem ser encontradas em: https://tools.ietf.org/id/draft-gharris-opsawg-pcap-00.html

Usando a linguagem de programação Python (versão 3.X) desenvolva um programa que leia um arquivo capturado pelo tcpdump (alguns exemplos encontram-se disponíveis no Moodle) e responda:

a) Em que momento inicia/termina a captura de pacotes?
b) Qual o tamanho do maior pacote capturado?
c) Há pacotes que não foram salvos nas suas totalidades? Quantos?
d) Qual o tamanho médio dos pacotes capturados?
e) Qual o par de IP com maior tráfego entre eles?
f) Com quantos outros IPs o IP da interface capturada interagiu?

4. Fazer um programa que efetue a leitura dos arquivos que estão contidos no arquivo **serie_historica_anp.rar**[1] e realize as seguintes operações (descompacte o arquivo em um diretório/pasta chamado **serie_historica_anp**):

a) **NÃO** deverá ser utilizada a biblioteca CSV, PANDAS ou similares;
b) O programa deverá reconhecer a inclusão de novos arquivos no diretório/pasta sem ser necessária a alteração no código fonte.
c) O programa deverá criar um diretório chamado **dados_estatisticos** na mesma pasta que está o arquivo **.py**;
d) Deverá ser gerada uma lista contendo as seguintes informações de todos os arquivos lidos: **Regiao – Sigla**, **Estado – Sigla**, **Produto**, **Data da Coleta**, **Valor de Venda**, **Bandeira**;
e) A lista gerada deverá ser salva em um único arquivo chamado **serie_historica_anp.txt** dentro do diretório criado no item **a**. Cada informação deverá ser separada por **;** (ponto e vírgula).
f) Com base na lista gerada no item **b**, deverão ser geradas as seguintes listas abaixo (cada lista deverá ser salva em formato de arquivo com o dados separados por **;** no diretório criado no item **a**). O nome de cada arquivo está no início de cada tópico:
    i. Nome do arquivo: **media_bandeira.txt**
        bandeira – produto – ano – valor_medio_venda – quantidade_postos
    ii. Nome do arquivo: **media_produto_regiao.txt**
        produto – região – ano – valor_medio – quantidade_postos

[1] Dados extraídos de https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-abertos/serie-historica-de-precos-de-combustiveis (acessado em 17/01/2022)

5. Para realizar essa atividade será necessário efetuar o download do arquivo **dados_cartola_fc.rar** (disponível no Moodle).

   Desenvolver um programa que atenda aos seguintes requisitos:

   a) O programa deverá solicitar ao usuário o ano em que se deseja acessar os dados do Cartola FC;
   b) Uma vez informado o ano, o programa deverá abrir o arquivo correspondente. Lembre de tratar possíveis exceções que venham a surgir;
   c) 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

   d) Para cada esquema tático, deve-se selecionar a quantidade de jogadores por posição obedecendo a tabela do item **c** dessa questão;
   e) 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 x 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;