Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Informática

Disciplina: Programação Funcional
Professor: Nilton Luiz Queiroz Junior

Trabalho Prático 2

Instruções:

- A aplicação deverá ser implementada na linguagem Racket;
- O trabalho vale de 0,0 a 10,0 e corresponde a 100% da nota da segunda avaliação periódica;
- Serão usados como critérios de avaliação:
  - A corretude do programa com relação ao que foi pedido;
  - A indentação do código;
  - A nomeação das variáveis e funções;
- O trabalho deverá ser entregue via classroom;
  - A data de entrega será definida em sala;
- O nome do arquivo enviado deve seguir o padrão:
  - Nome1-RAXXXXXX_Nome-RAYYYYYY.zip;
  - O formato para submissão deve ser .zip, .rar, ou .tar;
- O trabalho poderá ser feito em duplas;

Descrição: Deverá ser desenvolvida uma aplicação que seja capaz de fazer a análise de dados transferidos pela rede considerando um arquivo de log

- As informações deverão ser armazenadas numa estrutura de memória principal;
- Serão passados dois argumentos para o programa:
  - O arquivo de entrada:
    - Esse arquivo possui os dados de transferências de pactoes de rede realizados. Ele é composto por 3 “colunas” serparadas pelo caractere de tabulação (“\t”):
      - O endereço fonte (ou seja, o endereço de IP que enviou os dados para a rede);
      - O endereço destino (ou seja, o endereço de IP para o qual o pactoe foi enviado);
      - O total de bytes transferidos (ou seja, o tamanho do pacote de dados enviado);
    - Exemplo de um arquivo de entrada:
      - Devido ao uso de tabulação, as colunas não estarão perfeitamente alinhadas,
      - No exemplo, cada coluna esta sublinhada, e as tabulações não.

```
10.1.1.5        95.101.25.133       128
95.101.25.133   10.1.1.5            66
10.1.1.5        200.16.49.85        80
10.1.1.5        95.101.25.133       45
200.16.49.85    10.1.1.5            66
205.217.29.196  10.1.1.5            196
```

- O ip a ser monitorado está no formato de ipv4 (quatro números inteiros de 0 a 255 separados pelo caractere “.”).
  - Exemplos:
    - 10.1.1.5
    - 192.168.1.3
    - 142.251.129.228
    - 186.233.154.103
- O arquivo de saída:
  - Após a execução do programa, esse arquivo deverá conter um resumo da quantiade de dados enviados de cada endereço fonte para cada um dos endereços destino, ordenado pelos endereços com a maior quantidade de bytes trasnferência;
  - Alguns endereços irão aparecer como fonte de algumas transerencias e destino em outras, nesses casos, não se deve acumular as quantidades.
    Exemplo:

    Suponha um arquivo que possui essas linhas:

    1. 10.1.1.2 10.1.1.3 10
    2. 10.1.1.2 10.1.1.3 15
    3. 10.1.1.2 10.1.1.3 5
    4. 10.1.1.5 10.1.1.2 20
    5. 10.1.1.5 10.1.1.2 10
    6. 10.1.1.5 10.1.1.2 30
    7. 10.1.1.3 10.1.1.2 40
    8. 10.1.1.3 10.1.1.2 10

    O endereço 10.1.1.2 aparece como fonte nas 3 primeiras linhas e como destino nas demais. Dessa forma, as contagens de transferências devem ser separadas. Sendo assim, nesse caso a saída seria algo do tipo (mudando apenas o texto de saída):

    Origem: 10.1.1.5 Destino: 10.1.1.2 Bytes transferidos: 60 bytes
    Origem: 10.1.1.3 Destino: 10.1.1.2 Bytes transferidos: 50 bytes
    Origem: 10.1.1.2 Destino: 10.1.1.3 Bytes transferidos: 30 bytes

    Observe que mesmo ocorrendo transferências de 10.1.1.2 para 10.1.1.3, e de 10.1.1.3 para 10.1.1.2, elas são "separadas" na saída.

- Um arquivo contendo código para realizar o “parsing” dos argumentos passados para o programa, a leitura do arquivo de entrada (que gera uma única string) e a escrita do arquivo de saída (que recebe uma string para escrever no arquivo) está

disponível no classroom da disciplina. Além disso, também estão disponíveis alguns arquivos de entrada e as respostas esperadas para cada um deles.

- Para a ordenação **não é permitido que a dupla utilize funções prontas em racket, ou seja, deverá ser implementado algum algoritmo de ordenação.**
- As demais funções para manipulação de listas, tais como concatenação, remoção, entre outras, podem ser usadas.
- Dicas:

  Para facilitar o desenvolvimento, é sugerido que a dupla consulte as seguintes funções da linguagem racket (além, é claro, daquelas vistas em sala):

  - (string-split string): Quebra a string em uma lista de acordo com um separador (por padrão, o separador é dado pelos caracteres de espaço em branco). Link para a documentação:
    https://docs.racket-lang.org/reference/strings.html#%28def._%28%28lib._racket%2Fstring..rkt%29._string-split%29%29
  - (string-append str1 str2 …): Retorna a concatenação de todas as strings passadas como argumento. Link para a documentação:
    https://docs.racket-lang.org/reference/strings.html#%28def._%28%28quote._~23~25kernel%29._string-append%29%29
  - (string->number string): avalia a string e a trasnforma a em um número. Link para a documentação:
    https://docs.racket-lang.org/reference/generic-numbers.html#%28def._%28%28quote._~23~25kernel%29._string-~3enumber%29%29
  - E outras funções de comparação e manipulação de strings;
  - Para que sejam escritas novas linhas no arquivo de texto, utilize o caractere '\n' na string de saída.
  - Para facilitar a criação da string de saída utilize a função format:
    (format str v_1 v_2 ...): formata a string str colocando os valores v_i nos locais desejados para a string. Link para a documentação:
    https://docs.racket-lang.org/reference/Writing.html#%28def._%28%28quote._~23~25kernel%29._format%29%29

PROBLEMAS COM TRABALHOS COPIADOS:

Quem copiar o trabalho, da internet ou de outra dupla, terá o trabalho anulado (zerado). Quem conceder a cópia também.