UFF – Universidade Federal Fluminense
TIC – Instituto de Computação
TCC – Departamento de Ciência da Computação

TCC 00.308 | Programação de Computadores I | Turma A-1 | 2023.2
Professor Leandro Augusto Frata Fernandes

Verificação Suplementar (VS) | Aplicada em 20/12/2023

INSTRUÇÕES GERAIS

1. A avaliação é individual e sem consulta.
2. A única linguagem de programação aceita é Python 3.
3. Só é permitido o uso de folhas fornecidas pelo professor.
4. Questões sem desenvolvimento são consideradas incorretas.
5. As questões podem apresentar restrições em seu enunciado. Será atribuída nota ZERO para a questão caso alguma de suas restrições não seja cumprida.
6. Os programas escritos devem funcionar para qualquer caso de entrada possível definido pelo enunciado, não apenas para o(s) exemplo(s) dado(s).
7. O(A) estudante não pode sair da sala durante a avaliação. Caso o faça, então deve encerrar a prova e entregar as soluções produzidas até o momento.
8. Conversas e trocas de qualquer tipo de mensagem verbal ou textual são proibidas durante a avaliação e são consideradas tentativa de “cola”.
9. É proibido o uso de smartphone, tablet, notebook ou qualquer outros dispositivos eletrônicos durante a prova, sendo o acesso aos mesmos considerado tentativa de “cola”.
10. Será atribuída nota ZERO aos estudantes que: (i) forem pegos tentando “colar”; (ii) forem pegos tentando “passar cola”; (iii) que tenham “colado”, ou (iv) que tenham “passado cola”. A coordenação do curso será notificada do ocorrido para tomar as providências cabíveis.

QUESTÃO 1 - Mensagens de MasterApp

MasterApp é um aplicativo concorrente do WhatsApp. Ele faz a mesma coisa que seu rival, exceto sucesso.

Surgiu uma oportunidade para você se juntar ao time de desenvolvimento e mostrar suas habilidades de programador(a) Python. A tarefa que lhe foi passada é resolver o problema de ordenação das mensagens enviadas para um grupo qualquer. A versão atual do MasterApp mostra no grupo as mensagens conforme elas vão chegando. O caso é que a lentidão na rede pode fazer com que mensagens se atrasem, causando confusão, pois para um usuário as mensagens chegam em uma ordem enquanto que para outro usuários chegam em outra ordem e, com isso, a conversa vira uma bagunça. A solução para esse problema é reter em arquivos texto as mensagens enviadas por cada membro do grupo por um tempo para só depois exibi-las por ordem de envio e não por ordem de chegada.

Entrada
A entrada consiste em um conjunto de arquivos texto. Um arquivo chamado “`grupo.txt`” possui uma lista de códigos identificadores e nomes de usuários que formam um grupo. Cada usuário aparece em uma linha do arquivo, sendo cada linha formada por uma chave alfanumérica (o código identificador), seguida por um espaço em branco e pelo nome do usuário. O nome do usuário por ter qualquer tamanho e quaisquer caracteres, exceto quebra de linha.

Cada usuário do grupo possui um arquivo texto próprio cujo nome é “`código-identificador.txt`”, onde “`código-identificador`” é substituído pelo código do usuário. Dentro desses arquivos, cada linha contém uma data e hora de envio no formato “`hora:minuto:segundo dia/mês/ano`”, seguido por um caractere “`@`” e pela mensagem em si. As mensagens podem ter qualquer tamanho e quaisquer caracteres, exceto quebra de linha.

Todos os arquivos utilizam UTF-8 e quebra de linha no padrão adotado pelo Linux.

Saída

A saída consiste em um arquivo texto chamado "`mensagens.txt`" que contém em cada linha uma das mensagens enviadas para o grupo. As mensagens devem ser colocadas em ordem cronológica e envio no formato "`[nome-do-usuário] mensagem`", onde "`nome-do-usuário`" deve ser substituído pelo nome do usuário que enviou a mensagem e "`mensagem`" pela mensagem em si. Não existem datas e horas iguais entre mensagens enviadas para o grupo.

Exemplos

Arquivo `grupo.txt`

```
9hty67 Cicrano da Silva
Tp891h Beltrano Moraes
UuUi12 Fulano de Oliveira
```

Arquivo `9hty67.txt`

```
17/12/2023 09:06:12@Partiu!
17/12/2023 09:08:00@Tá me zuando?
```

Arquivo `Tp891h.txt`

```
17/12/2023 09:07:10@Por quê?
17/12/2023 09:05:33@Fala aí, galera. Bora pra praia?
```

Arquivo `UuUi12.txt`

```
17/12/2023 09:10:05@Não é isso não. Fiquei em VS em Prog I e tenho que estudar :(
17/12/2023 09:06:45@Ih, não vai dar.
```

Arquivo `mensagens.txt`

```
[Beltrano Moraes] Fala aí, galera. Bora pra praia?
[Cicrano da Silva] Partiu!
[Fulano de Oliveira] Ih, não vai dar.
[Beltrano Moraes] Por quê?
[Cicrano da Silva] Tá me zuando?
[Fulano de Oliveira] Não é isso não. Fiquei em VS em Prog I e tenho que estudar :(
```

Restrições

1) Seu programa deve funcionar com qualquer entrada que siga o formato especificado, não só para o exemplo dado no enunciado.

2) Deve ser utilizada subprogramação na ordenação das mensagens e na leitura e escrita de arquivos.

3) A ordenação deve ser feita pela implementação de um dos três algoritmos de ordenação vistos em sala de aula. Logo, não é permitido o uso do método `.sort`, da função `sorted` ou similares.

4) Não é permitido alterar o conteúdo das variáveis `sys.stdin` e `sys.stdout`.

5) Não é permitido utilizar bibliotecas como `pandas`, `numpy` ou similares, nem banco de dados.

6) Não é permitido o uso do módulo `datetime` ou de módulos similares.

Distribuição de Pontos

Esta questão vale 7,0 dos 10,0 pontos da avaliação. Os pontos são distribuídos em:

Q1.1 - Abertura e leitura dos arquivos de entrada conforme especificação: 1,0

Q1.2 - Abertura e escrita do arquivo de saída conforme especificação: 1,0
Q1.3 - Organização do fluxo do programa e da informação pelo uso de variáveis e listas adequadas: 2,0
Q1.4 - Ordenação conforme especificado: 2,0
Q1.5 - Uso correto e adequado de subprogramas: 1,0

QUESTÃO 2 - Meu Comando Split

Vixe! Parece que a instalação do ambiente de desenvolvimento Python está com problema e o comando `split` de `str` parou de funcionar. Por causa disso, você deverá implementar sua própria divisão de string em palavras. A assinatura da função requerida é:

```
meu_split(texto)
```

onde o argumento `texto` é do tipo `str` e o retorno esperado é uma lista com as palavras contidas em `texto`, na mesma ordem em que lá aparecem. Entende-se por palavras as sequências de caracteres entre o início da string, espaços em branco e final da strings. Ou seja, a função `meu_split` tem o mesmo comportamento de `texto.split(' ')`.

Entrada
A primeira linha da entrada informa o valor inteiro N (0 ≤ N ≤ 100), que indica a quantidade de conjuntos de teste existentes. As próximas N linhas representam, cada uma, um conjunto de teste composto por uma única string contendo de 0 (zero) a 20 (vinte) palavras.

Saída
Para cada conjunto de teste, o programa deve imprimir na saída padrão uma linha com o texto "`Teste t`", onde `t` é a numeração do caso de teste e que vai de 1 a N. Em seguida, devem ser impressas as palavras contidas na string do conjunto de teste atual, uma por linha. No final da saída de cada conjunto de teste deve ser impressa uma linha em branco.

Exemplos

Entrada	Saída
3 Abacaxi BANANA 123 Existem quatro espaços aqui	Teste 1 Abacaxi BANANA 123 Teste 2 Teste 3 Existem quatro espaços aqui

Restrições
1) Não é permitido o uso da função `split` nativa da API da linguagem Python, nem de funções similares importadas de qualquer módulo pré-existente.

2) É preciso definir a função `meu_split` exatamente como especificado no enunciado. Esta função não faz leitura nem escrita na entrada e saída padrão.

Distribuição de Pontos
Esta questão vale 3,0 dos 10,0 pontos da avaliação. Os pontos são distribuídos em:

Q2.1 - Leitura das entradas e escrita das saídas no formato especificado: 1,0
Q2.2 - Implementação da função `meu_split` conforme especificação: 2,0