Organização e Recuperação de Dados

1º Trabalho Prático

O 1º trabalho prático da disciplina **Organização e Recuperação de Dados** consiste na construção de um programa a ser desenvolvido em conformidade com as especificações abaixo. O programa deverá ser escrito na linguagem Python e poderá ser feito em equipes de até 3 alunos.

A equipe deverá enviar o código fonte pelo *Google Classroom* em um arquivo nomeado com os nomes dos integrantes da equipe. **O trabalho deverá ser apresentado pela equipe em data e horário agendado pelo professor.**

Especificação

O arquivo *filmes.dat* possui informações sobre filmes. Os dados dos filmes estão armazenados em registros de tamanho variável, em formato similar ao utilizado nas aulas práticas. O arquivo possui um cabeçalho de 4 bytes (um número inteiro com sinal) e os campos de tamanho dos registros têm 2 bytes (um número inteiro sem sinal). Cada filme possui um identificador único que servirá como chave primária, o qual segue o campo de tamanho.

Busca, Inserção e Remoção

Dado o arquivo *filmes.dat*, o seu programa deverá oferecer as seguintes funcionalidades principais:

- Busca de um filme pelo ID;
- Inserção de um novo filme;
- Remoção de um filme.

As operações a serem realizadas em determinada execução serão especificadas em um arquivo de operações, o qual será passado ao programa por parâmetro. Dessa forma, **o programa não possuirá interface com o usuário** e executará as operações na sequência em que estiverem especificadas no arquivo de operações.

A execução do arquivo de operações será acionada pela linha de comando, no seguinte formato:

```
$ python programa.py -e arquivo_operacoes
```

sendo `-e` a flag que sinaliza o modo de execução e `arquivo_operacoes` o nome do arquivo que contém as operações a serem executadas. Para simplificar o processamento do arquivo de operações, considere que ele sempre será fornecido corretamente (i.e., o seu programa não precisa verificar a integridade desse arquivo).

Formato do Arquivo de Operações

O arquivo de operações deve possuir uma operação por linha, codificada com o identificador da operação (b = *busca*, i = *inserção* ou r = *remoção*) e respectivos argumentos. A seguir é exemplificado o formato de um arquivo de operações.

```
b 20
i 66|500 Dias com Ela|Marc Webb|2009|Comédia, Drama, Romance|95|Joseph Gordon|
r 153
r 230
i 11|O Exorcista|William Friedkin|1973|Terror, Drama|122|Ellen Burstyn, Max von Sydow, Linda
Blair| i 150|O Tigre e o Dragão|Ang Lee|2000|Ação, Aventura, Fantasia|120|Chow Yun|
```

O arquivo acima representa a execução consecutiva das seguintes operações:

- Busca pelo registro de chave 20
- Inserção do registro do filme de identificador 66 (“500 Dias com Ela”)
- Remoção do registro de chave 153
- Remoção do registro de chave 230
- Inserção do registro do filme de identificador 11 (“O Exorcista”)
- Inserção do registro do filme de identificador 150 (“O Tigre e o Dragão”)

Com base no arquivo de operações mostrado acima, o programa deverá apresentar a seguinte saída:

```
Busca pelo registro de chave "20"
20|Forrest Gump|Robert Zemeckis|1994|Drama, Romance|142|Tom Hanks, Robin Wright, Gary Sinise (93 bytes)

Inserção do registro de chave "66" (77 bytes)
Local: fim do arquivo

Remoção do registro de chave "153"
Registro removido! (92 bytes)
Local: offset = 477 bytes (0x1dd)

Remoção do registro de chave "230"
Erro: registro não encontrado!

Inserção do registro de chave "11" (97 bytes)
Local: fim do arquivo

Inserção do registro de chave "150" (77 bytes)
Tamanho do espaço reutilizado: 92 bytes
Local: offset = 477 bytes (0x1dd)
```

Gerenciamento de Espaços Disponíveis

As alterações que venham a ocorrer no arquivo *filmes.dat* deverão ser persistentes. A remoção de registros será lógica e o espaço resultante da remoção deverá ser inserido na Lista de Espaços Disponíveis (LED). **A LED deverá ser mantida no próprio arquivo** e os ponteiros da LED devem ser gravados como números inteiros de 4 bytes com sinal. O seu programa deverá implementar todos os mecanismos necessários para o gerenciamento da LED e reutilização dos espaços disponíveis utilizando a estratégia **melhor ajuste (*best-fit*)**.

No momento da inserção de novos registros, a LED deverá ser consultada. Se existir um espaço disponível para a inserção, o novo registro deverá ser inserido nesse espaço. Sobras de espaço resultantes da inserção poderão permanecer como fragmentação interna. Caso não seja encontrado na LED um espaço adequado para o novo registro, ele deverá ser inserido no final do arquivo.

Impressão da LED

A funcionalidade de impressão da LED também será acessada via linha de comando, no seguinte

formato: `$ python programa.py -p`

sendo `-p` a flag que sinaliza o modo de impressão. Sempre que ativada, essa funcionalidade apresentará na tela os *offsets* dos espaços disponíveis que estão encadeados na LED, iniciando pela cabeça da LED. Veja abaixo um exemplo de como seria feita a impressão supondo que há três espaços disponíveis no arquivo:

```
LED -> [offset: 1850, tam: 90] -> [offset: 477, tam: 92] -> [offset: 1942, tam: 109] -> [offset:
-1] Total: 3 espacos disponiveis
```

Note que o arquivo *filmes.dat* deve existir para que o seu programa execute. Caso o arquivo não exista, o programa deve apresentar uma mensagem de erro e terminar.

Compactação do arquivo

A funcionalidade de compactação do arquivo *filmes.dat* também será acessada via linha de comando, no seguinte formato:

```
$ python programa.py -c
```

sendo `-c` a flag que sinaliza o modo de compactação. Sempre que ativada, essa funcionalidade deverá gerar uma nova versão do arquivo *filmes.dat* na qual os espaços disponíveis tenham sido fisicamente removidos. A compactação deverá remover a fragmentação externa, não sendo necessário eliminar a fragmentação interna. Uma vez compactado, o arquivo deverá estar pronto para novas rodadas de execuções.

BOM TRABALHO!