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 2 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(a) professor(a).

Especificação

O arquivo *dados.dat* possui informações sobre jogos. Os dados dos jogos estão armazenados em registros de tamanho variável, em formato similar ao utilizado nas aulas práticas. O arquivo possui um cabeçalho de 4 bytes e os campos de tamanho dos registros têm 2 bytes. Cada jogo possui os seguintes campos:

1. IDENTIFICADOR do jogo (utilizado como chave primária);
2. TÍTULO;
3. ANO;
4. GÊNERO;
5. PRODUTORA;
6. PLATAFORMA.

Busca, Inserção e Remoção

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

- Busca de um jogo pelo IDENTIFICADOR;
- Inserção de um novo jogo;
- Remoção de um jogo.

As operações a serem realizadas em determinada execução serão especificadas em um arquivo de operações, o qual será passado ao programa como um 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 22
i 147|Resident Evil 2|1998|Survival horror|Capcom|PlayStation|
r 99
r 230
i 181|Pac-Man|1980|Maze|Namco|Arcade|
i 144|The Sims|2000|Life simulation|Electronic Arts|PC|
```

O arquivo acima representa a execução consecutiva das seguintes operações:

- Busca pelo registro de chave 22
- Inserção do registro do jogo de identificador 147 (“Resident Evil 2”)

- Remoção do registro de chave 99
- Remoção do registro de chave 230
- Inserção do registro do filme de identificador 181 (“Pac-Man”)
- Inserção do registro do jogo de identificador 144 (“The Sims”)

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

```
Busca pelo registro de chave "22"
22|Tetris|1984|Puzzle|Elorg|Electronika 60| (43 bytes)

Inserção do registro de chave "147" (60 bytes)
Local: fim do arquivo

Remoção do registro de chave "99"
Registro removido! (94 bytes)
Local: offset = 6290 bytes (0x1892)

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

Inserção do registro de chave "181" (35 bytes)
Tamanho do espaço reutilizado: 94 bytes (Sobra de 57 bytes)
Local: offset = 6290 bytes (0x1892)

Inserção do registro de chave "144" (53 bytes)
Tamanho do espaço reutilizado: 57 bytes
Local: offset = 6327 bytes (0x18b7)
```

Gerenciamento de Espaços Disponíveis

As alterações que venham a ocorrer no arquivo *dados.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. O seu programa deverá implementar todos os mecanismos necessários para o gerenciamento da LED e reutilização dos espaços disponíveis utilizando a estratégia **pior ajuste (*worst-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 deverão ser reinseridas na LED, a menos que sejam menores do que um determinado limiar (p.e., 10 bytes). 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: 4, tam: 80] -> [offset: 218, tam: 50] -> [offset: 169, tam: 47] -> [offset: -1]
Total: 3 espacos disponiveis
```

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

BOM TRABALHO!