Exercício 01: Implemente uma função que simula o comando **grep** do Unix. A função receberá dois parâmetros:

- um arquivo texto com registros codificados usando \n como delimitador de de registros, e | como delimitador de campos;
- uma string de consulta que deseja-se verificar sua existência e ocorrências no arquivo;

A saída da função é um conjunto com todos os registros onde a informação foi encontrada no arquivo texto.

Uma sugestão de assinatura da função/método (em python) é:

```
def grep(arquivo, string)
```

Exercício 02: Crie uma nova função baseada no exercício anterior e retorne agora todos os registros e índices das linhas onde existe a string consultada.

Exercício 03: Escreva uma função para leitura de registros de tamanho variável usando RRN (*Relative Record Number* - Número de Registro Relativo), que retorna um registro baseado em sua posição no arquivo. Por exemplo: se for solicitado para recuperar o 50º registro do arquivo, a função deve se deslocar por 49 registros e retornar/imprimir apenas o conteúdo do 50º registro. Uma sugestão de assinatura da função/método (em python) é:

```
def readRecordByRRN(file, RNN)
```

Algumas observações:

- Use o dataset de Animes da aula passada;
- Use arquivos cujos **registros** possuem **tamanho fixo**, mas os **campos** têm **tamanhos variados**;
- O tamanho do registro é o tamanho do maior registro do dataset. Descubram esses valores dinamicamente;
- Completem os espaços não utilizados do registro imprimindo algum caractere especial de sua escolha (*, -, .);
- No acesso direto, a primeira linha do arquivo possui RRN = 0. Assim, se a função for chamada com os seguintes parâmetros:

  ```
  readRecordByRRN(arquivo, 0),
  ```

  ela deve retornar o primeiro registro do arquivo. Se RRN = 1, retorna o segundo registro. Se RRN = 2, retorna o terceiro, e assim por diante …