O objetivo aqui é te dar a oportunidade de trabalhar como você trabalharia no dia a dia, usando os teus programas (editor/terminal/navegador/etc). Fique à vontade para pesquisar soluções e documentações que possam te ajudar.

Leia com atenção e boa sorte! E lembre-se, **o código deve ser escrito em Ruby**.

Para fazer este teste, escreva o código no seu editor de preferência e responda este email com os arquivos de código em anexo.

O email com as respostas deve ser enviado no máximo 3 horas após o recebimento deste email.

Não se preocupe se achar que o tempo é curto para todas as questões, o objetivo é esse mesmo. Faça o que conseguir e, se estiver no meio de uma questão quando o tempo acabar, entregue o que tiver escrito até então.

Questão 1

O objetivo desta questão é escrever um método que receba uma lista de hashes e retorne um string.

Considere o seguinte código.

```
input = [
  { name: 'Maria Neusa de Aparecida',
    cpf: '97905796671',
    state: 'Sao Paulo',
    value: '1234' },
  { name: 'Ricardo Fontes',
    cpf: '44010762900',
    state: 'Rio Grande do Sul',
    value: '567' }
]

def solucao(arg)
  # Retornar um string
end

solucao(input) == "Maria Neusa97905796671Sao Paulo  1234       \nRicardo Fon44010762900Rio Grande 567        "
```

A sua tarefa é preencher o conteúdo do método `solucao` de modo que:

- Ele retorne um string que deve conter uma linha para cada elemento do array `input`
- Cada linha deve ser constituída do valor dos campos name, cpf, state e value, nessa ordem. Cada um dos valores deve ocupar 11 caracteres, truncando ou preenchendo com espaços quando necessário.
- O método deve funcionar mesmo com argumentos diferentes do exemplo acima. A única garantia é que será sempre uma lista de hashes com as chaves name, cpf, state e value.

Questão 2

O objetivo desta questão é extender a solução acima de modo que o formato do string retornado seja configurável.

Considere um arquivo yaml no seguinte modelo. As chaves de primeiro nível (name, cpf, etc) representam atributos do hash e **não são fixas** (quer dizer, elas podem diferir de um arquivo yaml para outro). As chaves de segundo nível são sempre as mesmas: `length`, `align`, e `padding`.

`length`
: Com quantos caracteres deve ser formatado esse campo no output. Se o conteúdo do campo ultrapassar esse comprimento, ele deve ser truncado.

`align`
: Indica se o conteúdo do campo deve ser alinhado a esquerda ou a direita.

`padding`
: Caso o conteúdo do cammpo seja mais curto que a `length` especificada, o `padding` indica se o espaço restante deve ser preenchido com espaços em branco ou zero.

Exemplo yaml 1

```
# format-1.yaml
cpf:
  length: 11
  align: left
  padding: spaces
name:
  length: 14
  align: left
  padding: spaces
value:
```

```
  length: 8
  align: right
  padding: zeroes
```

Utilizando esse arquivo com o exemplo da questão anterior, o retorno esperado seria:

```
"97905796671Maria Neusa de00001234\n44010762900Ricardo Fontes00000567"
```

Exemplo yaml 2

```
# format-2.yaml
cpf:
  length: 14
  align: right
  padding: zeroes
value:
  length: 8
  align: right
  padding: zeroes
state:
  length: 14
  align: left
  padding: spaces
```

Utilizando esse arquivo com o exemplo da questão anterior, o retorno esperado seria:

```
"00097905796671000012 34Sao Paulo     \n00044010762900000000567Rio Grande do "
```

Nesta solução, além de receber a lista de hashes como argumento, o método `solucao` deve ler estas configurações de um arquivo yaml de nome “format.yaml”.

Se preferir, fique a vontade para usar uma class ou module ao invés de só um método.

Questão 3

O objetivo desta questão é implementar a operação inversa da questão 2.

Isto é: escreva um método que recebe como argumento um string e retorna um hash. A forma como ele deve ler o string para extrair os valores é dada por um arquivo yaml como os da questão 2.

Por exemplo, se seguirmos o arquivo “format-1.yaml”.

```
cpf:
  length: 11
  align: left
  padding: spaces
name:
  length: 14
  align: left
  padding: spaces
value:
  length: 8
  align: right
  padding: zeroes
```

Um método corretamente implementado deve se comportar da seguinte forma:

```
solucao("97905796671Maria Neusa de00001234") == { cpf: '97905796671', name: 'Maria Neusa de', value: '1234' }
solucao("44010762900Ricardo Fontes00000567") == { cpf: '44010762900', name: 'Ricardo Fontes', value: '567' }
```