Operações com Strings: Codificação em **Base85**

Neste exercício, você colocará em prática alguns conhecimentos de operações bit a bit, manipulação de strings e de codificação através da representação de uma string ASCII em **Base85**.

Base85 é um grupo de esquemas de codificação binária para texto que representa uma sequência binária como uma string ASCII por meio da tradução para uma representação *radix-85*.

Sua vantagem em relação a outros esquemas de codificação que provêem o mesmo tipo de funcionalidade é que sua representação é mais compacta. Por exemplo, enquanto que para cada 3 caracteres o esquema de codificação *Base64* necessita de 4 caracteres codificados em uma string ASCII imprimível, o **Base85** codifica 4 caracteres em 5. Ou seja, enquanto o overhead do *Base64* é de 1 caractere para cada 3 caracteres de mensagem, o overhead do **Base85** é de 1 caractere para cada 4 caracteres.

Na prática, existem 95 caracteres da tabela ASCII que são imprimíveis. Para o **Base85** (variante *Z85* ou *ZeroMQ*), estamos interessados em apenas 85 desses caracteres conforme listagem abaixo:

```
- 0 - 9: 0 1 2 3 4 5 6 7 8 9
- 10 - 19: a b c d e f g h i j
- 20 - 29: k l m n o p q r s t
- 30 - 39: u v w x y z A B C D
- 40 - 49: E F G H I J K L M N
- 50 - 59: O P Q R S T U V W X
- 60 - 69: Y Z .  : + = ^ ! /
- 70 - 79: * ? & < > ( ) [ ] {
- 80 - 84: } @ % $ #
```

Para codificar uma mensagem, a implementação deve tomar 4 bytes (ou 4 octetos) por vez e convertê-los em 5 caracteres imprimíveis. Os quatro octetos deverão ser tratados como um número de 32 bits sem sinal (formato *big endian*). Os 5 caracteres deverão ser impressos de acordo com a ordem em que primeiro toma-se o caractere mais significativo para depois apresentarem-se os caracteres menos significativos (formato *big endian*).

Um exemplo de codificação:

Entrada: `"Alo "`

- Entrada em binário (hexadecimal, **com espaço ao final e sem as aspas delimitadoras**): `41 6c 6f 20`
- Entrada em decimal: 1097625376
- Entrada representada por potências de 85: $21 \times 85^4 + 2 \times 85^3 + 25 \times 85^2 + 39 \times 85 + 61$
- Para gerar a saída, pegamos cada número em negrito e consultamos na tabela de caracteres imprimíveis o representante em código.
  - Por exemplo, 21 mapeará em `l`.

Saída: `l2pDZ`

Requisitos

Sua aplicação deverá receber em entrada em console uma string de até 32 bytes/caracteres e calcular a representação dessa string em **Base85**.

Sua aplicação deverá rejeitar entradas cujo comprimento em bytes/caracteres não seja múltiplo de 4.

Entrada

A entrada é composta por strings ASCII de até 32 bytes/caracteres de comprimento.

Saída

A saída é a representação em **Base85** da string apresentada na entrada ou a string `Entrada invalida.` em caso de entrada inválida.

Exemplo de Entrada (com espaço após o 'o')

```
Alo
```

Exemplo de Saída

```
l2pDZ
```

Exemplo de Entrada

```
Alo
```

Exemplo de Saída

```
Entrada invalida.
```

Exemplo de Entrada

```
Alo m
```

Exemplo de Saída

```
Entrada invalida.
```

Exemplo de Entrada

```
Alo mundo!!!
```

Exemplo de Saída

```
l2pDZzfo-bzY{nE
```

Exemplo de Entrada

```
Man is distinguished, not only b
```

Exemplo de Saída

```
o<}]Zx(+zcx(!xgzFa9aB7/b}efF?GBrCHty<vdj
```

Author: Tiago Alves