

Problema: Filiação

O formato *CSV*, **C**omma-**S**eparated-**V**alues, é um tipo de arquivo que armazena dados com valores separados por vírgula. Cada linha do arquivo, ao abrir utilizando programas como Microsoft Excel ou Google Sheets, será transformada em uma linha na planilha, e cada coluna na planilha terá o dado que está separado por vírgula. Por exemplo, suponha que você esteja montando uma tabela onde a primeira coluna represente um nome de uma pessoa, a segunda coluna o nome da mãe e a terceira coluna o nome do pai de uma pessoa:

| | | |
|------------------------|------------------|-------------------|
| Pedro da Silva Pereira | Adriana da Silva | Guilherme Pereira |
| Luiz Coelho | | Tomas Coelho |

O *CSV* correspondente seria:

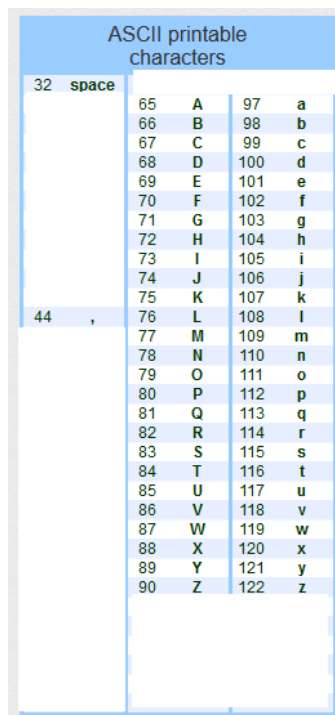
```
Pedro da Silva Pereira ,Adriana da Silva ,Guilherme Pereira
Luiz Coelho , ,Tomas Coelho
```

Faça uma função chamada *separaLinhaCSV*, que recebe uma string, que representa o nome completo de uma pessoa (nome e sobrenome, separados por espaço), o nome completo da mãe e o nome completo do pai de uma pessoa, e retorne uma estrutura do tipo *tipoFiliacao*, que possui informações de nome, nome da mãe e nome do pai. Para este problema, não há a necessidade de realizar validações. Considere que os nomes possuem apenas letras de A à Z, maiúsculo e minúsculos, e espaços.

```
struct tipoFiliacao separaLinhaCSV(char linha[240]);
```

Cada argumento da função representa, respectivamente:

- Os nome da pessoa, da mãe e do pai respectivamente, da forma **<PESSOA>**,**<MAE>**,**<PAI>**. Considere que este argumento possui somente caracteres apresentados na Figura 1.



| ASCII printable characters | | | |
|----------------------------|-------|-----|---|
| 32 | space | | |
| 65 | A | 97 | a |
| 66 | B | 98 | b |
| 67 | C | 99 | c |
| 68 | D | 100 | d |
| 69 | E | 101 | e |
| 70 | F | 102 | f |
| 71 | G | 103 | g |
| 72 | H | 104 | h |
| 73 | I | 105 | i |
| 74 | J | 106 | j |
| 75 | K | 107 | k |
| 44 | , | 108 | l |
| 77 | M | 109 | m |
| 78 | N | 110 | n |
| 79 | O | 111 | o |
| 80 | P | 112 | p |
| 81 | Q | 113 | q |
| 82 | R | 114 | r |
| 83 | S | 115 | s |
| 84 | T | 116 | t |
| 85 | U | 117 | u |
| 86 | V | 118 | v |
| 87 | W | 119 | w |
| 88 | X | 120 | x |
| 89 | Y | 121 | y |
| 90 | Z | 122 | z |

Figura 1: Caracteres possíveis para o primeiro argumento da função.

A estrutura *tipoFiliacao* deve ser composta obrigatoriamente por:

- Um vetor de caracter chamado **nome**, com 80 posições, que indica o nome completo da pessoa, que possui caracteres que pertencem a tabela ASCII;
- Um vetor de caracter chamado **nomeMae**, com 80 posições, que indica o nome completo da mãe, que possui caracteres que pertencem a tabela ASCII e;
- Um vetor de caracter chamado **nomePai**, com 80 posições, que indica o nome completo do pai, que possui caracteres que pertencem a tabela ASCII.

Restrições

A criação de outras funções auxiliares é permitida. Porém, para este exercício, não é permitido o uso da biblioteca *string.h*

Ao enviar a sua solução pro MOJ, envie somente o arquivo com a extensão C com a função exigida do enunciado, as estruturas, a inclusão das bibliotecas utilizadas e as funções auxiliares (caso existam). Não inclua neste arquivo a função *main*.

Entrada

A sua função não fará nenhuma leitura da entrada padrão.

Saída

A sua função não fará nenhuma impressão na saída padrão.

Exemplos

Exemplo 1

Suponha que a função seja chamada da seguinte forma:

```
separaLinhaCSV("Pedro da Silva Pereira ,Adriana da Silva ,Guilherme Pereira ");
```

A função deve retornar a estrutura preenchida da seguinte forma:

| nome | nomeMae | nomePai |
|------------------------|------------------|-------------------|
| Pedro da Silva Pereira | Adriana da Silva | Guilherme Pereira |

Exemplo 2

Suponha que a função seja chamada da seguinte forma:

```
separaLinhaCSV("Luiz Coelho , ,Tomas Coelho ");
```

A função deve retornar a estrutura preenchida da seguinte forma:

| nome | nomeMae | nomePai |
|-------------|---------|--------------|
| Luiz Coelho | | Tomas Coelho |

Problema: Descendence

The file format *CSV*, **C**omma-**S**eparated-**V**alues, is a type of file that stores data splited by comma. Each line of this file, when opened using programs such as Microsoft Excel or Google Sheet, will be transform in a line of the spreadsheet, and each column of the spreadsheet will have the data that is separeted by comma. For example, suposes that you are creating a table that the first column indicates a person's name, the second column his mother and the third column his father:

| | | |
|------------------------|------------------|-------------------|
| Pedro da Silva Pereira | Adriana da Silva | Guilherme Pereira |
| Luiz Coelho | | Tomas Coelho |

The *CSV* will be:

```
Pedro da Silva Pereira ,Adriana da Silva ,Guilherme Pereira
Luiz Coelho , ,Tomas Coelho
```

Write a function called *separaLinhaCSV*, that receives a string, that holds the information of the complete name (name and last name, separated by space), the complete name of the mother and the complete name of the father, and returns a structure of *tipoFiliacao*'s type, that holds the information of the person's name, the person's mother name and the person's father name. For this problem, there is no need to validate the string. Considers that the names has only letters from A to Z, uppercase and lowercase, and spaces.

```
struct tipoFiliacao separaLinhaCSV(char linha[240]);
```

Each argument of the function represents, respectively:

- The names of the person, of the mother and of the father, seperated by comma, as the template **<PERSON>**,**<MOTHER>**,**<FATHER>**. Consider that this argument only has characters as show at the Figure 1.

| ASCII printable characters | | | |
|----------------------------|-------|-----|---|
| 32 | space | | |
| 65 | A | 97 | a |
| 66 | B | 98 | b |
| 67 | C | 99 | c |
| 68 | D | 100 | d |
| 69 | E | 101 | e |
| 70 | F | 102 | f |
| 71 | G | 103 | g |
| 72 | H | 104 | h |
| 73 | I | 105 | i |
| 74 | J | 106 | j |
| 75 | K | 107 | k |
| 44 | , | 108 | l |
| 77 | M | 109 | m |
| 78 | N | 110 | n |
| 79 | O | 111 | o |
| 80 | P | 112 | p |
| 81 | Q | 113 | q |
| 82 | R | 114 | r |
| 83 | S | 115 | s |
| 84 | T | 116 | t |
| 85 | U | 117 | u |
| 86 | V | 118 | v |
| 87 | W | 119 | w |
| 88 | X | 120 | x |
| 89 | Y | 121 | y |
| 90 | Z | 122 | z |

Figura 2: Possible characters.

The structure *tipoFiliacao* must have the following elements:

- A character's array called **nome**, with size of *80*, that indicates the complete name of the person, that has only characters in ASCII table;
- A character's array called **nomeMae**, with size of *80*, that indicates the complete name of the mother's person, that has only characters in ASCII table and;
- A character's array called **nomePai**, with size of *80*, that indicates the complete name of the father's person, that has only characters in ASCII table.

Restrictions

It is allowed to write other auxiliary functions. However, it is not allowed the use any of the functions from *string.h*.

To submit your solution to MOJ, send only the C file with the function that is request and the auxiliary functions (if they exist). Do not includes in this file the function *main*.

Input

There is no input for this problem.

Output

There is no output for this problem.

Examples

Example 1

Suppose that your function is called as in:

```
separaLinhaCSV("Pedro da Silva Pereira ,Adriana da Silva ,Guilherme Pereira ");
```

Therefore, your function must return a filled structure as follow:

| nome | nomeMae | nomePai |
|------------------------|------------------|-------------------|
| Pedro da Silva Pereira | Adriana da Silva | Guilherme Pereira |

Exemplo 2

Suppose that your function is called as in:

```
separaLinhaCSV("Luiz Coelho , ,Tomas Coelho ");
```

Therefore, your function must return a filled structure as follow:

| nome | nomeMae | nomePai |
|-------------|---------|--------------|
| Luiz Coelho | | Tomas Coelho |