Observação: é expressamente proibido a utilização de qualquer função da biblioteca `string.h` ou de qualquer outra biblioteca em C que faça a manipulação de *strings*.

01. Desenvolva um programa em C que receba um texto fornecido pelo usuário e apresente a quantidade de caracteres alfabéticos contidos nesse texto. Por exemplo:

```
Entrada: 15 abacates
Saída: 8

Entrada: quinze (15) abacates
Saída: 14
```

02. Implemente em C a função `int isNullY(char *string)` que verifica se a *string* passada como parâmetro é uma *string* nula, ou seja, não contém nenhum elemento, devolvendo o valor lógico correspondente. Apresente também um programa de testes para validar a função desenvolvida.

```
isNullY("")          -     1 (true)
isNullY("Fulano")    -     0 (false)
```

03. Implemente em C a função `void strcpyY(char *destino, char *origem, int limit)` que copie a *string* de origem para a *string* de destino, até que o delimitador da *string* seja atingido ou o limite estabelecido. Apresente também um programa de testes para validar a função desenvolvida.

```
Entrada: "", "Fulano"
Saída: "Fulano", "Fulano"
```

04. Implemente em C a função `void strcatY(char *destino, char *origem, int limit)` que concatene a *string* de origem no final da *string* de destino, até que o delimitador da *string* de origem seja atingido ou o limite estabelecido. Apresente também um programa de testes para validar a função desenvolvida.

```
Entrada: "Fulano", " da Silva"
Saída: "Fulano da Silva", " da Silva"
```

05. Implemente em C a função `int strlenY(char *string, int limit)` que retorne a quantidade de caracteres presentes na *string* passada como parâmetro, ou `limit` caso a *string* ultrapasse o limite estabelecido. Apresente também um programa de testes para validar a função desenvolvida.

```
Entrada: "Fulano da Silva"
Saída: 15
```

06. Implemente em C a função `void initY(char *string)` que inicialize a *string* passada como parâmetro como uma *string* nula. Apresente também um programa de testes para validar a função desenvolvida.

```
Entrada: "Fulano da Silva"
Saída: ""
```

07. Implemente em C a função `int indexOfY(char *string, char ch, int limit)` que retorne a posição da primeira ocorrência do caractere `ch` na *string*. Caso a *string* não contenha o caractere desejado, a função deverá retornar -1. Apresente também um programa de testes para validar a função desenvolvida.

```
Entrada: "Fulano da Silva", 'a'
Saída: 3
```

08. Implemente em C a função `int lastIndexOfY(char *string, char ch, int limit)` que retorne a posição da última ocorrência do caractere `ch` na *string*. Caso a *string* não contenha o caractere desejado, a função deverá retornar -1. Apresente também um programa de testes para validar a função desenvolvida.

```
Entrada: "Fulano da Silva", 'a'
Saída: 14
```

09. Implemente em C a função `void toLowerCaseY(char *string, int limit)` que coloque todos os caracteres de *string* para minúsculas. Apresente também um programa de testes para validar a função desenvolvida.

```
Entrada: "Fulano da Silva"
Saída: "fulano da silva"
```

10. Implemente em C a função `void toUpperCaseY(char *string, int limit)` que coloque todos os caracteres de *string* para maiúsculas. Apresente também um programa de testes para validar a função desenvolvida.

```
Entrada: "Fulano da Silva"
Saída: "FULANO DA SILVA"
```

11. Implemente em C a função `void replaceY(char *string, char old, char new, int limit)` que substitua todos os caracteres `old` da *string* pelos caracteres `new`. Apresente também um programa de testes para validar a função desenvolvida.

```
Entrada: "Fulano da Silva", 'a', 'x'
Saída: "Fulxno dx Silvx"
```

12. Implemente em C a função `void replaceFirstY(char *string, char new, int size, int limit)` que substitua os primeiros caracteres da *string* por `new`. A quantidade de caracteres a serem substituídos é fornecido por `size`. Apresente também um programa de testes para validar a função desenvolvida.

```
Entrada: "Fulano da Silva", 'x', 3
Saída: "xxxano da Silva"
```

13. Implemente em C a função `void replaceLastY(char *string, char new, int size, int limit)` que substitua os últimos caracteres da *string* por `new`. A quantidade de caracteres a serem substituídos é fornecido por `size`. Apresente também um programa de testes para validar a função desenvolvida.

```
Entrada: "Fulano da Silva", 'x', 3
Saída: "Fulano da Sixxx"
```

14. Implemente em C a função `int equalsY(char *string1, char *string2, int limit)` que verifique se as duas strings são iguais. Apresente também um programa de testes para validar a função desenvolvida.

```
equalsY("Fulano", "Fulano")           -     1 (true)
equalsY("Fulano", "Ciclano"     )     -     0 (false)
```

15. Implemente em C a função `int equalsIgnoreCaseY(char *string1, char *string2, int limit)` que verifique se as duas strings são iguais, ignorando se os caracteres estão em maiúsculas ou minúsculas. Apresente também um programa de testes para validar a função desenvolvida.

```
equalsIgnoreCaseY("Fulano", "FULANO")       -     1 (true)
equalsIgnoreCaseY("Fulano", "Ciclano" )     -     0 (false)
```

16. Desenvolva um programa em C que leia o nome completo do usuário do teclado e o escreva novamente na tela no formato sobrenome, nome. Por exemplo:

```
Nome: João Carlos Cunha
Cunha, João Carlos
```

17. Implemente em C a função `void trimY(char *string, int limit)` que elimine todos os espaços em branco a esquerda e a direita da palavra. Apresente também um programa de testes para validar a função desenvolvida.

```
Entrada: "   Fulano    da    Silva   "
Saída: "Fulano    da    Silva"
```

18. Implemente em C a função `void trimAllY(char *string, int limit)` que elimine todos os espaços em branco a esquerda e a direita da palavra, bem como os espaços em branco repetidos dentro da palavra. Apresente também um programa de testes para validar a função desenvolvida.

```
Entrada: "   Fulano    da    Silva   "
Saída: "Fulano da Silva"
```

19. Implemente em C a função `void duplicaY(char *string, int limit)` que duplique o texto contido dentro da *string*. Apresente também um programa de testes para validar a função desenvolvida.

```
Entrada: "Fulano"
Saída: "FulanoFulano"
```

20. Implemente em C a função `void worduprY(char *string, int limit)` que coloque a primeira letra de cada palavra em maiúscula e as restantes em minúsculas. Supõe-se que a separação entre palavras é realizada por espaços em branco. Apresente também um programa de testes para validar a função desenvolvida.

```
Entrada: "ERA uma vez"
Saída: "Era Uma Vez"
```