1 Estrutura String (+++)

 (+++)

Faça um programa que implemente um tipo de dado para manipular *strings* como uma estrutura. Para isso você deverá declarar o tipo **String** e implementar as funções listadas no código abaixo. A declaração das funções devem ser deduzidas a partir do programa principal dado.

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>


typedef struct {
    // ???
} String;

/**
 * Função que cria e retorna uma nova String a partir de um texto constante.
 * A memória alocada deve se ajustar à quantidade de caracteres do texto,
 * incluindo também o caracter de término de string '\0'.
 *
 * @param PARAMETROS
 * @return ?
 */
TIPO string_new( PARAMETROS );

/**
 * Função que libera a memória de uma String e atribui NULL a seu ponteiro.
 * Exemplo de chamada dessa função:
 *   String * s = string_new("teste");
 *   string_free( &s );
 *   // neste ponto s tem seu valor igual a NULL. Ou sejam a função string_free
 *   // deve atualizar o conteúdo de s.
 * @param PARAMETROS
 * @return ?
 */
TIPO string_free( PARAMETROS );

/**
 * Função que cria e retorna uma nova String lida pelo terminal.
 * @param PARAMETROS
 * @return ?
 */
TIPO string_read( PARAMETROS );


/**
 * Função que cria e retorna uma nova String com todas as letras em maiúsculo
 * a partir de uma String passada como parâmetro.
 * @param PARAMETROS
 * @return ?
 */
TIPO string_upper( PARAMETROS );


/**
 * Função que cria e retorna uma nova String formada pela concatenação de 2 Strings
   passadas como parâmetros.
```

```
 * @param PARAMETROS
 * @return ?
 */
TIPO string_concat( PARAMETROS );


/**
 * Função que cria e retorna uma nova String formada por uma substring de uma String passada por parâmetro.
 * @param str ponteiro para uma String
 * @param i índice do caracter inicial
 * @param n quantidade de caracteres a partir o índice inicial
 * @return ?
 */

TIPO string_substr( String * str, int i, int n );

/**
 * Função que cria e retorna uma nova cópia de  uma String passada por parâmetro.
 * @param PARAMETROS
 * @return ?
 */
TIPO string_copy( PARAMETROS );

/**
 * Função retorna a posição da primeira ocorrência de um caracter em uma String a partir do indice i.
 * @param str ponteiro para uma String
 * @param i índice do início da busca pelo caractere
 * @param c caractere a ser encontrado
 * @return índice do caracter. -1 caso o caractere não exista em str.
 */
TIPO string_find( String * str, int i, char c );

/**
 * Função que cria e retorna um vetor de substrings separadas por um caracter.
 * Cada elemento do vetor é um ponteiro para String. Exemplo:
 *      String * s = string_new("adeus mundo cruel!");
 *      int n;
 *      String ** vs = string_split( s, ' ', &n);
 *
 *      // O resultado deve ser:
 *          // vs[0]->str deve conter "adeus"
 *          // vs[1]->str deve conter "mundo"
 *          // vs[2]->str deve conter "cruel"
 *          // n == 3
 *
 * @param str ponteiro para uma String
 * @param c caracter separador
 * @param n quantidade de strings
 * @return ?
 */
TIPO string_split( String * str, char c, int * n );

int main() {

    String *a = NULL, *b = NULL, *c = NULL;
    int n, i;
    String ** vs;

```

```
109     printf("Criando string a:\n");
110     a = string_new("Isso e um texto qualquer!");
111     string_info(a);
112 
113     //printf("Criando string b:\n");
114     b = string_read();
115     string_info(b);
116 
117     //printf("Concatenando:\n");
118     c = string_concat(a, b);
119     string_info(c);
120     string_free(&c);
121 
122     //printf("Upper:\n");
123     c = string_upper(b);
124     string_info(c);
125     string_free(&c);
126 
127     //printf("Substring 0[3]:\n");
128     c = string_substr(a, 0, 3);
129     string_info(c);
130     string_free(&c);
131 
132     //printf("Substring 4[4]:\n");
133     c = string_substr(a, 4, 4);
134     string_info(c);
135     string_free(&c);
136 
137     string_copy( &c, a );
138     string_info(c);
139 
140 
141     vs = string_split(a, ' ', &n);
142 
143     for( i = 0; i < n; i++ ) {
144         string_info( vs[i] );
145     }
146 
147     string_free(&a);
148     string_free(&b);
149     string_free(&c);
150 
151     return 0;
152 }
```

Entrada

Saída

Observações

Exemplo

Entrada	Saída