INF1007 - PROGRAMAÇÃO II

LISTA DE EXERCÍCIOS 3

1. Implemente uma função `inverte` que recebe como parâmetro uma cadeia de caracteres e retorna uma nova cadeia, que corresponde à cadeia original, lida de trás para frente (a cadeia passada como parâmetro não pode ser alterada). Por exemplo, recebendo como parâmetro a cadeia "Aluno", essa função retornaria a cadeia "onulA". Sua função deve ter o seguinte protótipo:

   ```
   char *inverte(char *s);
   ```

   Em seguida, escreva a função principal do programa e utilize a função `inverte` para inverter uma cadeia de caracteres informada pelo usuário.

2. Implemente a função `ultimo_nome` que recebe como parâmetro uma cadeia com o nome completo de uma pessoa e retorna um ponteiro (o endereço inicial) do último nome encontrado.

   Sua função deve obedecer o seguinte protótipo:

   ```
   char* ultimo_nome(char *nome_completo);
   ```

   Atenção: você não pode criar novas cadeias de caracteres dentro da função (estáticas ou dinâmicas).

   Exemplos:

   - Para "Rafael de Moura Machado" a sua função deve retornar um ponteiro para "Machado";
   - Para "Jose Silva" a sua função deve retornar um ponteiro para "Silva";
   - Para "ZeNinguem" a sua função deve retornar um ponteiro para "ZeNinguem";

   Considere que há apenas um espaço entre os nomes e que após o último nome não há espaços.

   Implemente também a função `verificaSobrenome`, que recebe duas cadeias com os nomes completos de duas pessoas, e verifica se essas duas pessoas têm o mesmo sobrenome (isto é, o mesmo último nome). Em caso positivo, a função deve retornar uma nova cadeia alocada dinamicamente com esse sobrenome. Em qualquer outro caso, a função deve retornar uma cadeia vazia. Havendo problema de alocação de memória, a função deve retornar NULL.

Para simplificar, considere que todas as letras são maiúsculas, sem acentos e sem cedilhas. Considere também que há apenas um único espaço em branco entre nomes e que após o último nome não há espaços em branco.

Para simplificar ainda mais, sempre que possível utilize as funções da biblioteca `string.h`:

```
int strlen (char* s);
int strcmp (char* s, char *t);
char* strcpy (char* destino, char* fonte);
char* strncpy (char* destino, char* fonte, int n);
char* strcat (char* destino, char* fonte);
```

Exemplos de nomes:

- Para “STEVEN PAUL JOBS” e “JOSE JOBS”, retorna “JOBS”;
- Para “JOSE JOBS” e “CHICO ANYSIO”, retorna cadeia vazia “”;
- Para “SILVA” e “SILVA”, retorna “SILVA”;
- Para “” e “”, retorna cadeia vazia “”;

Em seguida, crie a função principal do programa e utilize a função `verificaSobrenome` para verificar se os sobrenomes de dois nomes informados pelo usuário são iguais ou não. Use os exemplos acima para testar o seu programa.

3. Implemente uma função que receba como parâmetros uma cadeia de caracteres e um caractere. A função deve retirar da cadeia todas as ocorrências desse caractere. A função deve obedecer ao seguinte protótipo:

```
char* retiraChar(char *str, char c);
```

Você deve alocar a nova cadeia de caracteres dinamicamente e copiar os caracteres da cadeia original para a área de memória alocada, exceto os caracteres que deverão ser retirados da cadeia original. A cadeia alocada deve ter o tamanho exato para comportar os caracteres que serão copiados, além do caractere nulo.

Em seguida, escreva a função principal do programa que permita ao usuário digitar um texto (tamanho máximo 100). Após digitar o texto, o programa deve perguntar se o usuário deseja remover algum caractere do texto. Se o usuário digitar algum caractere diferente de 0, o programa deve usar a função `retiraChar` para eliminar o caractere digitado e exibir o resultado. O processo de eliminação de caracteres deve ser executado até que o usuário digite o caractere 0.

4. Implemente uma função recursiva para comparar duas cadeias de caracteres. A função deve retorna 1 se as duas cadeias forem iguais ou 0 se elas forem diferentes. A função deve obedecer ao seguinte protótipo:

   ```
   int compara_r(char *str1, char *str2);
   ```

   Escreva também uma função recursiva para procurar um caractere em uma cadeia de caracteres. Caso o caractere seja encontrado, a função deve retornar 1, caso contrário, deve retornar 0. A função deve obedecer ao seguinte protótipo:

   ```
   int busca_r(char *str, char c);
   ```

   IMPORTANTE: Ambas as funções devem ser implementadas de forma recursiva.

   Em seguida, escreva a função principal do programa e utilize a função `compara_r` para comparar duas strings fornecidas pelo usuário (tamanho máximo 30) e escrever na tela se elas são iguais ou não. Em seguida, o usuário deve digitar um caractere e programa deve utilizar a função `busca_r` para verificar se esse caractere existe em uma ou em ambas as strings fornecidas pelo usuário.

5. O cadastro dos motoristas de uma empresa de transportes utiliza um vetor de inteiros (`matricula[N]`) para armazenar o número da matricula dos seus motoristas e um segundo vetor de inteiros (`multas[N]`) para armazenar a quantidade de multas de cada motorista (onde `multas[k]` armazena o número de multas do motorista de `matricula[k]`).

   Implemente uma função recursiva `contaPioresMotoristas`, que recebe o número de motoristas da empresa, o vetor com as inscrições e o vetor com as multas dos motoristas, e retorna o número de motoristas com quantidade de multas >10. A função deve também exibir na tela as matriculas desses piores motoristas.

   IMPORTANTE: A função `contaPioresMotoristas` deve ser implementada de forma recursiva.

   Em seguida, crie a função principal do programa para verificar os piores motoristas utilizando a função `contaPioresMotoristas` e os seguintes vetores:

   ```
   int matricula[] = {2648, 2674, 4128, 9852, 2147, 6258, 3245}
   int multas[]    = {2, 0, 12, 3, 20, 15, 1}
   ```