Prova 3

Algoritmos e Estruturas de Dados I - turma TE

Professor: Pedro O.S. Vaz de Melo

30 de maio de 2014 (valor: 30 pontos)

Nome: _____

escrevendo o meu nome eu juro que seguirei o código de honra

Código de Honra para este exame:

- Não darei ajuda a outros colegas durante os exames, nem lhes pedirei ajuda;
- não copiarei nem deixarei que um colega copie de mim;
- não usarei no exame elementos de consulta não autorizados.

Informações importantes:

- Considere que todos os procedimentos e funções pedidas nesta prova serão implementados no módulo `prova3.h`.
- Em questões que pede um **programa**, este deve ser completo, com bibliotecas (incluindo o módulo `prova3.h` quando necessário), função main, etc. Se deve ser feita uma **função**, somente a função é suficiente. Se deve ser feito um **procedimento**, somente o procedimento é suficiente.
- A interpretação das questões da prova faz parte do critério de avaliação. Caso tenha dúvida sobre a sua interpretação de uma determinada questão, escreva as suas suposições na resolução da mesma.
- Vocês podem utilizar qualquer função pedida na prova em suas questões. Considere que a implementação da função que você está usando está correta.
- **Lembrete:** Toda *string* termina (e DEVE terminar) com o caractere `'\0'`.

Referências:

Função/Operador	Descrição	Exemplo
`FILE* fopen(const char *filename, const char *mode)`	abre o arquivo `filename` no modo `mode`	`FILE *temp = fopen("temp.txt", "w");`
`int fclose ( FILE * arq )`	fecha o arquivo `arq`	`fclose(arq);`
`int feof ( FILE * arq )`	verificar se o arquivo `arq` chegou ao fim	`int fim_arq = feof(arq);`
`int fprintf(FILE *arq, const char *format, valores/variáveis);`	escreve dados no arquivo `arq`	`fprintf (arq, "valor de aux: %d", aux);`
`void* malloc (size_t size);`	aloca um bloco de memória de tamanho `size`, retornando um ponteiro para o início do bloco.	`int *p1 = (int*)malloc(sizeof(int));`
`char* fgets (char *str, int num, FILE *arq)`	Lê uma linha do arquivo apontado por `arq` ou no máximo `num` caracteres	`fgets(buffer, 1000, arq);`
`char *strtok (char *str, const char *delimiters)`	Retorna um campo da *string* `str` separado por um dos caracteres contidos em `delimiters`. Se `str` é `NULL`, busca o campo da string usada na chamada anterior.	`char *nome = strtok(buffer, ",");`
`void free (void *p);`	Desaloca o bloco de memória apontado por `p`.	`free(p);`
`int rename(const char *old, const char *new);`	Renomeia o arquivo de nome `old` para o nome `new`. Retorna `-1` se um erro ocorrer.	`rename("dados.txt", "temp.txt");`
`int remove(const char *filename)`	Deleta o arquivo de nome `filename`	`remove("dados.txt");`

1. (*10 points*) Escreva uma função RECURSIVA que recebe um ponteiro `str` para uma *string* como parâmetro e retorna quantas palavras distintas existem no texto apontado por `str` . Considere que todas as palavras contidas no texto são divididas por espaços únicos. Exemplo: se `str` apontar para `"A bola rola, carambola!"` você deve retornar `4`. Sua função não pode usar *loops* (`for, while, etc`) nem variáveis globais e deve ter o seguinte protótipo:

```
int numPalavras(char *str);
```

2. (*10 points*) Escreva uma função de nome `retornaAbrev` que recebe um nome como parâmetro e retorna a sua abreviação. Assim, essa função recebe uma *string* `str` c como parâmetro e retorna uma nova *string* contendo a abreviação desse nome. O espaço alocado para armazenar a abreviação não deve ser maior nem menor que a própria abreviação (contando com o `'\0'`). Exemplo: se a *string* de entrada for `"Pedro Olmo Stancioli Vaz de Melo"`, você deve retornar `"P.O.S.V.d.M."`. A função deve ter o seguinte protótipo:

```
int* retornaAbrev(char *str);
```

3. (*10 points*) Um jogo *online* está preocupado com a privacidade e segurança de seus usuários e, por isso, resolveu abreviar todos os nomes dos jogadores contidos no seu arquivo de cadastro. Assim, escreva um programa que lê esse arquivo, cujo nome é *cadastro.dat*, e o modifica, convertendo todos os nomes para as suas respectivas abreviações. Use a função `retornaAbrev` do exercício anterior para fazer isso. Como o arquivo pode conter milhões de regitros, é **necessário** que você desaloque toda memória que você alocou e não for usar mais. Formato do arquivo:

```
login#senha#nome
```

Exemplo de arquivo:

```
tyrion#ihatecersei#Tyrion Lannister da Silva
davos#ilovestannis#Davos Seaworth Vasconcelos
dog#auau#Theon Greyjoy de Melo
```