Prova 3

Algoritmos e Estruturas de Dados I - turma TW

Professor: Pedro O.S. Vaz de Melo

27 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.

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 fscanf(FILE *arq, const char *format, endereço das variáveis);`	lê dados númericos do arquivo `arq`	`fscanf(arq, "%f", &nota1);`
`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. (*6 points*) Escreva uma função RECURSIVA que receba por parâmetro um valor inteiro e maior que zero x retorna a **magnitude** de x. A magnitude de um número x é um outro número, em escala de 10, que representa a grandeza de x. Intuitivamente falando, a magnitude é a maior potência de 10 contida no número. Ex: a magnitude de 8 é 1, a magnitude de 17 é 10, a magnitude de 189 é 100, a magnitude de 4631 é 1000 e assim por diante. Sua função não pode usar *loops* (`for, while, etc`) nem a função `log10` e deve ter o seguinte protótipo:

```
int magnitude(unsigned int x);
```

2. (*6 points*) Escreva uma função RECURSIVA que recebe um ponteiro para uma *string* como parâmetro e retorna o seu tamanho. Sua função não pode usar *loops* (`for, while, etc`) e deve ter o seguinte protótipo:

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

3. (*8 points*) Ao preencher formulários online, muitas vezes os usuários colocam diversos espaços em branco antes e depois do texto digitado. Apesar de parecer insignificante, isso pode gerar problemas em diversos processos como, por exemplo, de comparação de *strings* e de alocação de memória.
Assim, escreva um procedimento de nome `trim` que recebe uma *string* `str` como parâmetro **por referência** e a modifica removendo todos os espaços em branco que a precede e a sucede, caso necessário. Em suma, você deve transferir a *string* sem os espaços em branco iniciais e finais para uma nova área de memória (*heap*) e, após isso, fazer com que o parâmetro `str` aponte para essa nova área. Exemplo: se a `string` inicial for `"  AEDS1  "`, você a deve transformar para `"AEDS1"`. Não se esqueça de desalocar o espaço da *string* antiga e usar o terminador `\0`. A função deve ter o seguinte protótipo:

```
void trim(char **str);
```

4. (*10 points*) Um arquivo que alimenta um sistema de catálogo de telefones contém diversos dados incorretos. Muitos dos nomes escritos no arquivo contém espaços em branco os precedendo e/ou sucedendo (ex: `"  Pedro Olmo  "`). Assim, escreva um programa que lê esse arquivo, cujo nome é *catalogo.dat*, e o modifica, removendo todos os espaços em branco que precedem e sucedem os nomes nele escritos. Use a função `trim` do exercício anterior para fazer isso. Formato do arquivo:

```
nome|telefone
```

Exemplo de arquivo:

```
Tyrion Lannister#9876-1234
Davos Seaworth#7654-2345
Theon Greyjoy#8765-6789
Sandor Clegane#7654-5678
Khal Drogo#9876-0123
```