Prova 3

Algoritmos e Estruturas de Dados I

Professor: Pedro O.S. Vaz de Melo

29 de junho de 2017 (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.

Referências:

Função/Operador	Descrição	Exemplo
`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));`
`FILE* fopen(const char *filename, const char *mode)`	abre o arquivo `filename` no modo `mode`	`FILE *temp = fopen("temp.txt", "w");`
`int fscanf(FILE *arq, const char *format, &variáveis);`	lê dados númericos do arquivo `arq`	`fscanf(arq, "%f", &nota1);`
`int fprintf(FILE *arq, const char *format, vars);`	escreve dados no arquivo `arq`	`fprintf (arq, "valor de aux: %d", aux);`
`int fclose ( FILE * arq )`	fecha o arquivo `arq`	`fclose(arq);`

1. (*7 points*) Escreva uma função RECURSIVA que recebe um apontador para uma *string* como parâmetro e retorna o número de espaços contidos na *string*. Ex: para a *string* `ola, tudo bem?`, a sua função deve retornar 2. A função deve ter o seguinte protótipo:

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

2. (*8 points*)
Escreva um procedimento RECURSIVO de protótipo `void imp2Cont(int i, int n)` que imprime a contagem crescente e, logo depois, a contagem decrescente de i até n. Sua função não pode usar *loops* (`for, while, etc`). Exemplo: imp2Cont(2,5) imprime

```
2
3
4
5
4
3
2
```

3. (*5 points*) Escreva um programa para criar um arquivo de nome `asc2.txt` e escrever neste arquivo a tabela ASCII, do código 0 ao código 127, contendo o código do caractere (em notação decimal) e o caractere em si. Abaixo uma parte do arquivo que você deve gerar a partir do seu programa:

```
63 ?
64 @
65 A
66 B
67 C
```

4. (*10 points*) Preencha o código abaixo de função que lê uma matriz quadrada simétrica de um arquivo e a retorna em memória alocada dinamicamente. Lembre-se que em uma matriz simétrica `M[i][j] = M[j][i]` e que em uma matriz quadrada o número de linhas é igual ao número de colunas.

```
double ** le_matriz_simetrica(char *nome_arquivo, int tamanho)
{
  int i, j;
  FILE *fd = fopen(_____________________________________________________);
  if(!fd) abort(); //fecha o programa

  double **M = _____________________________________________________;
  if(M==NULL) abort(); //fecha o programa

  for(i = 0; _____________________________; _____________________________) {

    M[i] = _____________________________________________________;
    if(M==NULL) abort(); //fecha o programa

    for(j = 0; _____________________________; _____________________________) {

      fscanf(_____________________________________________________________);
    }
  }

  ____________________________________________________________;

  ____________________________________________________________;
}
```

O parâmetro `nome_arquivo` indica o nome do arquivo com os dados na matriz e `tamanho` indica o número de linhas na matriz.
Como a matriz é simétrica, o arquivo de entrada contém apenas a metade inferior da matriz. Em outras palavras, o arquivo contém apenas os elementos `M[i][j]` onde `i` $\geq$ `j`. Um exemplo de arquivo de entrada para `tamanho = 4` é dado abaixo:

```
1.2
2.4 1.3
8.2 9.1 6.2
8.3 6.5 8.2 1.0
```

Importante: sua função deve alocar a menor quantidade possível de memória para armazenamento da matriz. Em particular, sua função deve alocar espaço apenas para elementos `M[i][j]` onde `i > j`, da mesma forma como o arquivo de entrada armazena apenas a metade inferior da matriz.