Prova 3

Algoritmos e Estruturas de Dados I

Professor: Ítalo Cunha e Pedro Vaz de Melo

13 de dezembro de 2015 (valor: 25 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 fclose ( FILE * arq )`	fecha o arquivo `arq`	`fclose(arq);`

1. (*5 points*) Escreva uma função RECURSIVA que receba por parâmetro dois valores inteiros x e y e retorne o resultado de x^y. Sua função não pode usar *loops* (`for, while, etc`) nem a função `pow`. A função deve ter o seguinte protótipo:

```
int potencia(int x, int y);
```

2. (*5 points*) Escreva uma função RECURSIVA que receba por parâmetro um ponteiro para uma *string* e a imprime invertida. Se a sua função receber um ponteiro para a *string* "bola", a sua função deve imprimir "alob". Sua função não pode usar *loops* (`for, while, etc`). Dica: lembre-se de uma propriedade de *strings* que vetores de caracteres não necessariamente possuem. A função deve ter o seguinte protótipo:

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

3. (*5 points*)

```
void impAst(int n) {
    while(n>0) { printf("*"); n--; }
printf("\n");
}
```

Escreva um procedimento recursivo de protótipo `void imprimeTriangulo(i,n)` que imprime um triângulo lateral na tela formado por asteriscos. O triângulo deve começar imprimindo i asteriscos na primeira linha, $i+1$ na segunda, até n asteriscos na $(n-i)$ésima linha. Depois deve imprimir $n-1$ asteriscos, $n-2$, até i asteriscos na última linha. Dica: use o procedimento `impAst`. Exemplo: imprimeTriangulo(3,5) imprime

```
***
****
*****
****
***
```

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();
  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(___________________, ___________________, ___________________);
    }
  }

  ____________________________________________________________________;

  ____________________________________________________________________;
}
```

Onde 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 > 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
```

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.