Escola Superior de Tecnologia
Primeiro Trabalho

Disciplina: Programação de Computadores e Algoritmos
Professor: Ricardo Rios
Data da Entrega: 31.10.2017
Turma: BSI02_T01-2017_Q2

Aluno:_____ **Matrícula:** _____

Aluno:_____ **Matrícula:** _____

Observações: Este trabalho deve ser entregue em meio digital. Poderá ser feito por no máximo três pessoas. Cada questão deve possuir seu próprio programa, por exemplo, a questão 01 deve ser apresentada em um programa chamado "questao_01.c". Todas as questões devem estar em um diretório cujo nome deve ser o título do trabalho seguido dos números de matrícula dos integrantes do trio, por exemplo, "Segundo_Trabalho-0493829438_948473939_34857439". No diretório deve haver um arquivo texto (readme.txt) com informações para compilação e execução dos programas. Este trabalho deve ser enviado para o e-mail **rrios@uea.edu.br**, com o título do assunto "[LP2] Segundo Trabalho, o diretório deve ser compactado e anexado à mensagem.

Questões

1. (1,0 ponto) Implemente uma função que indique se um ponto (x, y) está localizado dentro ou fora de um retângulo. O retângulo é definido por seus vértices inferior esquerdo (x0, y0) e superior direito (x1, y1). A função deve ter como valor de retorno 1 (um), se o ponto estiver dentro do retângulo, e 0 (zero) caso contrário, obedecendo ao protótipo:

   int dentro_ret(int x0, int y0, int x1, int y1, int x, int y);

2. (1,0 ponto) Implemente uma função para testar se um número inteiro é primo ou não. Essa função deve obedecer ao protótipo a seguir e ter como valor de retorno 1 (um) se **n** for primo e 0 (zero) caso contrário.

   int primo(int n);

3. (0,5 ponto) Implemente uma função que retorne o n-ésimo termo da série de Fibonacci. A série de Fibonacci é dada por: 1 1 2 3 5 8 13 21…, isto é, os dois primeiros termos são iguais a 1 (um) e cada termo seguinte é a soma dos dois termos anteriores. Essa função deve obedecer ao protótipo:

   int fibonacci(int n);

4. (0,5 ponto) Implemente uma função que retorne a soma dos ***n*** primeiros números naturais ímpares. Essa função deve obedecer ao protótipo:

   int soma_impares(int n);

5. (1,0 ponto)Implemente uma função que retorne uma aproximação do valor de ***p***, de acordo com a fórmula de Leibniz:

$$\pi=\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}\ldots\right)$$

Isto é:

$$\pi \approx 4 \times \sum_{i=0}^{n} -\frac{1^i}{2\times i+1}$$

Essa função deve obedecer ao seguinte protótipo, em que ***n*** indica o número de termos que deve ser usado para avaliar o valor de :

double pi(int n);

6. (1,0 ponto) Implemente uma função que calcule as raízes de uma equação do segundo grau, do tipo $ax^2+bx+c = 0$. Essa função deve obedecer ao protótipo:

int raizes(float a, float b, float c, float * x1, float * x2);

Essa função deve ter como valor de retorno o número de raízes reais e distintas da equação. Se existirem raizes reais, seus valores devem ser armazenados nas variáveis apontadas por x1 e x2.

7. (0,5 ponto) Implemente uma função que calcule a área da superfície e o volume de uma esfera de raio **r**. Essa função deve obedecer ao protótipo:

void clac_esfera(float r, float * area, float * volume);

A área da superfície e o volume são dados, respectivamente, por $4r^2$ e $4r^3/3$.

8. (0,5 ponto) Implemente uma função que receba como parâmetro um vetor de números reais (**vet**) de tamanho ***n*** e retorne quantos números negativos estão armazenados nesse vetor. Essa função deve obedecer ao protótipo:

int negativos(int n, float * vet);

9. (0,5 ponto) Implemente uma função que receba como parâmetro um vetor de números inteiros (**vet**) de tamanho ***n*** e retorne quantos números pares estão armazenados nesse vetor. Essa função deve obedecer ao protótipo:

int pares(int n, int * vet);

10. (0,5 ponto) Implemente uma função que receba como parâmetro um vetor de números inteiros (**vet**) de tamanho ***n*** e inverta a ordem dos elementos armazenados nesse vetor. Essa função deve obedecer ao protótipo:

    void inverte(int n, int * vet);

11. (1,0 ponto) Implemente uma função que permita a avaliação de polinômios. Cada polinômio é definido por um vetor que contém seus coeficientes. Por exemplo, o polinômio de grau 2, $3x^2+2x+12$, terá um vetor de coeficientes igual a v[]={12, 2, 3}. A função deve obedecer ao protótipo:

    double avalia(double * poli, int grau, double x);

Onde o parâmetro **poli** é o vetor com os coeficientes do polinômio, **grau** é o grau do polinômio, e **x** é o valor para o qual o polinômio deve ser avaliado.

12. (1,0 ponto) Implemente uma função que calcule a derivada de um polinômio. Cada polinômio é representado como exemplificado na questão 11. A função deve obedecer ao protótipo:

    void deriva(double * poli, int grau, double * out);

Onde **out** é o vetor, de dimensão grau-1, no qual a função deve guardar os coeficientes do polinômio resultante da derivada.

13. (0,5 ponto) Implemente duas versões de uma função, seguindo as diferentes estratégias discutidas para alocar matrizes, que determine se uma matriz é simétrica quadrada ou não.

14. (0,5 ponto) Implemente uma função para somar, subtrair, multiplicar e dividir duas matrizes. Essas funções devem obedecer aos protótipos:

    a. **float ** somar(float ** matrizA, float ** matrizB);**

    b. **float ** subtrair(float ** matrizA, float ** matrizB);**

    c. **float ** multiplicar(float ** matrizA, float ** matrizB);**

    d. **float ** dividir(float ** matrizA, float ** matrizB);**