

Universidade Federal do ABC Disciplina: MCTA028 - Programação Estruturada Professor: Jesús P. Mena-Chalco		Avaliação: Prova 01 (40 pontos) Turma: Noturno – A1 e A2 Data: 25/10/2016
Nome completo:		RA:

Instruções:

- A prova tem a duração de 1h50min.
- É proibido o uso de qualquer aparelho ou recurso de processamento e/ou comunicação.
- Utilize a linguagem C para todas as questões (não utilize pseudocódigo).

1. Para cada pergunta selecione uma opção. Resposta correta +2, incorreta -1. [12 pontos]

<pre>int funcao1A(int n){ if (n == 0 n == 1) return 1; else return 2 * n * funcao1A(n - 1); }</pre>	Para n=4, a função devolve o inteiro: a) 24 b) 44 c) 96 d) 192 e) 384
<pre>int funcao2A(int x, int y){ return x - 10.5 * y; }</pre>	Para x=1 e y=2, a função devolve o inteiro: a) -18 b) -19 c) -20 d) -21 e) -22
<pre>int funcao3A(int x, int y){ if (x>y) return 10; return 20; return 30; }</pre>	Para x=10 e y=10, a função devolve o inteiro: a) 10 b) 20 c) 30 d) 20, 30 e) 10, 20, 30
<pre>int funcao4(int n){ return n/3*5/2-n; }</pre>	Para n=8, a função devolve o inteiro: a) -2 b) -3 c) -6 d) -7 e) -8
<pre>int funcao5(int n){ int i, soma=0; for (i=0; i<n; i+=2) soma += i; return i; }</pre>	Para n=7, a função devolve o inteiro: a) 6 b) 8 c) 10 d) 12 e) 20
<pre>#include <stdio.h> void main() { int a = 100; int b = 33; int *c = &a; a = b - *c; printf("%d %d %d\n", a, b, *c); }</pre>	O programa ao lado imprime: a) %d %d %d b) 100 33 100 c) -67 33 -67 d) 100 33 *100 e) -67 33 *-67

2. A seguinte função devolve um número aleatório inteiro [3 pontos]

```
int gerarNumero() {
    srand(time(NULL));
    return rand();
}
```

Crie uma função que permita gerar um número aleatório entre **a** e **b**, isto é, gerar um número inteiro aleatório no intervalo **[a, b]**

```
int gerarNumeroNoIntervalo(int a, int b) {
    srand(time(NULL));
    return rand()%(b-a+1) + a;
}
```

3. Crie uma versão iterativa da seguinte função. Considere **n** sempre maior ou igual a zero. [5 pontos]

```
int funcaoR1(int n) {
    if (n==0)
        return 0;
    else {
        if (n%2==0)
            return n + funcaoR1(n-1);
        else
            return funcaoR1(n-1);
    }
}
```

```
int funcaoIterativa (int n) {
    int i, soma=0;

    for (i=0; i<=n; i+=2)
        soma += i;

    return soma;
}
```

4. Crie uma função que permita verificar se dois vetores, $v[0..n-1]$ e $w[0..m-1]$ de números inteiros são iguais. Sua função deverá devolver 1 se ambos os vetores forem iguais, caso contrário deve devolver 0. **Use somente ponteiros.** Não use colchetes, isto é, não use “[” ou “]”. [5 pontos]

```
int vetoresIguais (int *v, int n, int *w, int m) {
    int i=0;

    while ( i<n && i<m && *(v+i)==*(w+i) )
        i++;

    if (i==n && i==m)
        return 1;
    else
        return 0;
}
```

5. O primorial de um número inteiro positivo n é o produto de todos os primos menores ou iguais a n . Crie uma função recursiva que, dado um número inteiro positivo, devolva o seu Primorial [5 pontos]

```
long int primorial (int n)

// Desafio dado em aula
```

6. O número de Euler pode ser calculado pela somatoria de infinitos termos como:

$$e = \sum_{i=0}^{\infty} \frac{1}{i!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \dots$$

A função `euler1` permite calcular o valor de **e** somando termos maiores ou iguais do que um **epsilon** (valor muito pequeno, por exemplo, 0.0000001).

Crie uma versão recursiva da função iterativa `euler1`. [5 pontos]

Observação: Se sua função recursiva precisar de mais de um parâmetro deverá indicar seus valores considerados na chamada a função.

```
double euler1(double epsilon) {
    double e=0;
    int i, d=1;
    for (i=1; 1.0/d>epsilon; i++) {
        e += 1.0/d;
        d *= i;
    }
    return e;
}

double eulerRec(double epsilon, int i, int d) {
    if (1.0/d>epsilon)
        return 1.0/d + eulerRec(epsilon, i+1, d*i);

    return 0;
}

double euler2(double epsilon){
    return eulerRec(epsilon, 1, 1);
}
```

7. O elemento **minimax** de uma matriz bidimensional é o menor elemento da linha que contém o maior elemento de uma matriz. Crie uma função para encontrar o elemento minimax de uma matriz **M** de números inteiros com **n** linhas e **m** colunas. [5 pontos]

```
int minimax (int n, int m, int M[n][m]) {
    int i, j, menor;
    int maior = M[0][0];
    int imaior;

    for (i=0; i<n; i++)
        for (j=0; j<m; j++)
            if (maior < M[i][j]) {
                maior = M[i][j];
                imaior = i;
            }

    menor = maior;

    for (j=0; j<m; j++)
        if (menor > M[imaior][j])
            menor = M[imaior][j];

    return menor;
}
```

Universidade Federal do ABC		Avaliação: Prova 01 (40 pontos)
Disciplina: MCTA028 - Programação Estruturada		Turma: Noturno – A1 e A2
Professor: Jesús P. Mena-Chalco		Data: 25/10/2016
Nome completo:		RA:

Instruções:

- A prova tem a duração de 1h50min.
- É proibido o uso de qualquer aparelho ou recurso de processamento e/ou comunicação.
- Utilize a linguagem C para todas as questões (não utilize pseudocódigo).

1. Para cada pergunta selecione uma opção. Resposta correta +2, incorreta -1. [12 pontos]

<pre>int funcao1B(int n){ if (n == 0 n == 1) return 1; else return 2 * n * funcao1A(n - 1); }</pre>	<p>Para n=3, a função devolve o inteiro:</p> <p>a) 4 b) 24 c) 44 d) 96 e) 192</p>
<pre>int funcao2B(int x, int y){ return x - 10.5 * y; }</pre>	<p>Para x=3 e y=2, a função devolve o inteiro:</p> <p>a) -14 b) -15 c) -16 d) -17 e) -18</p>
<pre>int funcao3B(int x, int y){ if (x<y) return 10; return 20; return 30; }</pre>	<p>Para x=30 e y=30, a função devolve o inteiro:</p> <p>a) 10 b) 20 c) 30 d) 20, 30 e) 10, 20, 30</p>
<pre>int funcao4(int n){ return n/3*5/2-n; }</pre>	<p>Para n=7, a função devolve o inteiro:</p> <p>a) -2 b) -3 c) -6 d) -7 e) -8</p>
<pre>int funcao5(int n){ int i, soma=0; for (i=0; i<n; i+=2) soma += i; return i; }</pre>	<p>Para n=9, a função devolve o inteiro:</p> <p>a) 6 b) 8 c) 10 d) 12 e) 20</p>
<pre>#include <stdio.h> void main() { double a = 10.5; double b = 3.3; double *c = &a; a = 2 * a; printf("%.1f %.1f %.1f\n", a, b, 2.0*(*c)); }</pre>	<p>O programa ao lado imprime:</p> <p>a) %.1f %.1f %.1f b) 10 3 2.0*10 c) 10.5 3.3 10.5 d) 21.0 3.3 42.0 e) 10.5 3.3 21.0</p>

2. O número de Euler pode ser calculado pela somatoria de infinitos termos como:

$$e = \sum_{i=0}^{\infty} \frac{1}{i!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \dots$$

A função `euler1` permite calcular o valor de **e** somando termos maiores ou iguais do que um **epsilon** (valor muito pequeno, por exemplo, 0.0000001).

Crie uma versão recursiva da função iterativa `euler1`. [5 pontos]

Observação: Se sua função recursiva precisar de mais de um parâmetro deverá indicar seus valores considerados na chamada a função.

```
double euler1(double epsilon) {
    double e=0;
    int i, d=1;
    for (i=1; 1.0/d>epsilon; i++) {
        e += 1.0/d;
        d *= i;
    }
    return e;
}
```

```
double eulerRec(double epsilon, int i, int d) {
    if (1.0/d>epsilon)
        return 1.0/d + eulerRec(epsilon, i+1, d*i);

    return 0;
}
```

```
double euler2(double epsilon){
    return eulerRec(epsilon, 1, 1);
}
```

3. A seguinte função devolve um número aleatório inteiro [3 pontos]

```
int gerarNumero() {
    srand(time(NULL));
    return rand();
}
```

Crie uma função que permita gerar um número aleatório entre **a** e **b**, isto é, gerar um número inteiro aleatório no intervalo **[a, b]**

```
int gerarNumeroNoIntervalo(int a, int b) {
    srand(time(NULL));
    return rand()%(b-a+1) + a;
}
```

4. Crie uma versão iterativa da seguinte função. Considere n sempre maior ou igual a zero. [5 pontos]

```
int funcaoR2(int n) {
    if (n==0)
        return 0;
    else {
        if (n%2==1)
            return n + funcaoR2(n-1);
        else
            return funcaoR2(n-1);
    }
}
```

```
int funcaoIterativa (int n) {
    int i, soma=0;

    for (i=1; i<=n; i+=2)
        soma += i;

    return soma;
}
```

5. Crie uma função que permita verificar se dois vetores, $v[0..n-1]$ e $w[0..m-1]$ de números inteiros são iguais. Sua função deverá devolver 1 se ambos os vetores forem iguais, caso contrário deve devolver 0. Use somente ponteiros. Não use colchetes, isto é, não use “[” ou “]”. [5 pontos]

```
int vetoresIguais (int *v, int n, int *w, int m) {
    int i=0;

    while ( i<n && i<m && *(v+i)==*(w+i) )
        i++;

    if (i==n && i==m)
        return 1;
    else
        return 0;
}
```

6. O primorial de um número inteiro positivo n é o produto de todos os primos menores ou iguais a n . Crie uma função recursiva que, dado um número inteiro positivo, devolva o seu Primorial [5 pontos]

```
long int primorial (int n)

// Desafio dado em aula
```

7. O elemento **minimax** de uma matriz bidimensional é o menor elemento da coluna que contém o maior elemento de uma matriz. Crie uma função para encontrar o elemento minimax de uma matriz M de números inteiros com n linhas e m colunas. [5 pontos]

```
int minimax (int n, int m, int M[n][m])

// Similar à solução da prova A (deve-se identificar o menor valor da coluna!)
```