## Primeira parte do teste final de Algoritmos e Estruturas de Dados 4 de fevereiro de 2022 16h00m - 16h55m

Responda a todas as perguntas no enunciado do teste. Justifique todas as suas respostas.

Nome:

N. Mec.: \_\_\_\_\_

4.0 1: No seguinte código,

```
#include <stdio.h>
int f(int x) { return x * x - 1; }
int g(int x) { return x / 3; }

int main(void)
{
   int c = 0;
   for(int i = 0; i <= 10; i++)
      if( f(i) && g(i) )
      {
       printf("i = %d\n",i);
      c += g(i);
    }
   printf("c = %d\n",c);</pre>
```

## Fórmulas:

- $\bullet \sum_{k=1}^{n} 1 = n$
- $\bullet \ \sum_{k=1}^{n} k = \frac{n(n+1)}{2}$
- $\bullet \ \sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$
- $\bullet \sum_{k=1}^{n} k^3 = \left(\frac{n(n+1)}{2}\right)^2$
- $\bullet \ \sum_{k=1}^{n} \frac{1}{k} \approx \log n$
- $n! \approx n^n e^{-n} \sqrt{2\pi n}$

- 1.0 a) para que valores da variável i é avaliada a função g(x)?
- 1.0 **b)** que valores de i são impressos?
- 1.0 c) que valor de c é impresso?

}

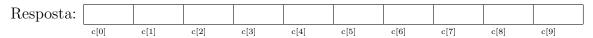
1.0 **d)** neste caso concreto, considerando que um int tem 32 *bits*, existe a possibilidade de *overflow* aritmético ao correr o programa?

4.0 2: Um programador inexperiente escreveu a seguinte função para copiar uma zona de memória com size bytes que começa no endereço src para uma outra zona de memória que começa no endereço dst.

```
void mem_copy(char *src,char *dst,size_t size)
{
  for(size_t i = 0;i < size;i++)
    dst[i] = src[i];
}</pre>
```

Responda às seguintes perguntas, considerando que para cada uma das duas primeiras o conteúdo **inicial** do array c é char c[10] = { 0,1,2,3,4,5,6,7,8,9 };

a) Qual o conteúdo do array c depois de mem\_copy(&c[4],&c[5],4); ter sido executado?



1.3 b) Qual o conteúdo do array c depois de mem\_copy(&c[5],&c[4],4); ter sido executado?

Resposta:										
	c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]	c[7]	c[8]	c[9]

1.4 c) Num dos casos anteriores a cópia do conteúdo de parte do array não foi feita corretamente; sugira uma maneira de corrigir este problema (não é obrigatório escrever código).
Resposta:

**4.0 3:** Ordene as seguintes funções por ordem crescente de ritmo de crescimento. Responda nas duas colunas da direita da tabela. Na coluna da ordem, coloque o número 1 na função com o ritmo de crescimento **menor** (e, obviamente, coloque o número 5 na com o ritmo de crescimento maior). Na coluna do termo dominante indique, usando a notação  $Big\ Oh$ , qual é o termo dominante; por exemplo, se na primeira coluna estivesse 3n+7, na segunda coluna deveria colocar  $\mathcal{O}(n)$ .

função	termo dominante	ordem
$42\frac{n^n}{n!}$		
$\sum_{k=1}^{n} \left( k^3 + \frac{1}{k} \right)$		
$\boxed{4n^4\log n^4 + 2022}$		
$1000n^3 + 1.001^n$		
$\frac{700}{n} + 300$		

**4.0 4:** A notação *big Oh* é usualmente usada para descrever a complexidade computacional de um algoritmo. Porquê?

Resposta (tente não exceder as 100 palavas):

**4.0 5:** Para a seguinte função,

```
int f(int n)
{
  int r = -20220204;

  for(int i = 0; i <= n; i++)
     for(int j = 2 * i; j <= 2 * n; j++)
        r += j / (i + 1);
  return r;
}</pre>
```

- 3.0  $(\bar{a})$  quantas vezes é executada a linha r += j / (i + 1);?
- 1.0 **b**) qual é a complexidade computacional da função?

