First part of the final exam of Algoritmos e Estruturas de Dados

Febuary 4, 2022 **16h00m – 16h55m**

Nome: ______________________________

N. Mec.: __________

Formulas:

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

4.0 **1:** In the following code,

```
#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);
}
```

1.0 **a)** the function `g(x)` is evaluated for which values of `i`?

1.0 **b)** which values of `i` are printed?

1.0 **c)** what is the final value of `c`?

1.0 **d)** in this particular case, assuming that an `int` has 32 bits, can arithmetic overflow occur?

4.0 **2:** An inexperient programmer wrote the following function to copy a memory area with `size` bytes that starts at `src` into another memory area that starts at `dst`.

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

Answer to the following question, considering in each of the following questions the **initial** contents of the `c` array is `char c[10] = { 0,1,2,3,4,5,6,7,8,9 };`

1.3 **a)** What is the final contents of the `c` array after the call `mem_copy(&c[4],&c[5],4);`?

Answer:

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

1.3 **b)** What is the final contents of the `c` array after the call `mem_copy(&c[5],&c[4],4);`?

Answer:

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

1.4 **c)** In one of the two previous cases the copy is not done correctly. Sugest a way to correct this problem.

Answer:

4.0 $\boxed{\mathbf{3:}}$ Order the following function is increasing order of rate of growth. Answer in the two columns on the right. The the order column, place the number in the row corresponding to the slowest growing function.

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

4.0 $\boxed{\mathbf{4:}}$ The *big Oh* is usually used to descrebe the computational complexity of an algorithm. Why?

4.0 **5:** For the following function,

```
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;
  return r;
}
```

3.0 **a)** how many times is the line `r += j / i;` executed?

1.0 **b)** what is the computational complexity of this functio?