Primeiro teste de **Algoritmos e Estruturas de Dados**

13 de Outubro de 2017 **Duração: 1 hora**

Justifique todas as suas respostas.

Nome:

N. Mec.:

Fórmulas:

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

3.0 **1:** No seguinte código,

```
#include <stdio.h>

int f(int x) { return 2 * x + 3; }
int g(int x) { return x * x - 7; }

int main(void)
{
  for(int i = -5;i <= 5;i++)
    if( (f(i) > 0) || (g(i) > 0) )
      printf("%d\n",i);
  return 0;
}
```

1.5 **a)** para que valores da variável `i` é avaliada a função `g(x)`?

1.5 **b)** que valores de `i` são impressos?

1.5 **2:** No seguinte código,

```
int a[10],*b = &a[7];
```

qual é o índice do elemento do array `a` que é referenciado por `b[-4]`?

4.0 **3:** A complexidade computacional de muitos algoritmos é expressa usando a notação "big Oh" ($\boldsymbol{O}$) em vez da notação "Big Theta" ($\boldsymbol{\Theta}$). Porquê? (Nota: dois terços da cotação para uma boa explicação das duas notações, um terço para uma boa explicação do porquê.)

3.0 **4:** Ordene as seguintes funções por ordem crescente de ritmo de crescimento. Responda nesta folha, usando o número das funções na sua resposta.

Número da função	função
1	$\frac{n!}{n^{100}} - 1$
2	$n \log n + \sqrt{n}$
3	$1.2^n + 17 + n^3$
4	$23 + \frac{\log n}{n}$
5	$n^4 + \frac{1000}{n}$

Resposta:

3.0 **5:** Para a seguinte função,

```
int f(int x)
{
  int i,j,r = 0;

  for(i = 0;i <= x;i++)
    for(j = i;j >= 0;j--)
      r += i - j;
  return r;
}
```

1.5 **a)** quantas vezes é executada a linha `r += i - j;`?

1.5 **b)** que valor é devolvido pela função?

4.0 **6:** O seguinte trecho de código reserva espaço para uma matriz com n linhas com uma determinada forma. Não é reservado espaço para os elementos da matriz fora dessa forma.

```
int n;    // the number of rows of the matrix
int **a;  // the matrix

void init_a(void)
{
  int i,k,s,*p; // auxiliary variables

  // the total number of elements of the matrix
  s = ;

  // allocate memory for the array of pointers
  a = (int **)malloc((size_t)n * sizeof(int *));
  // the memory for ALL elements
  p = (int *)malloc((size_t)s * sizeof(int));
  for(i = 0;i < n;i++)
  {
    // the number of valid elements on the i-th line
    k = 2 * i + 1;
    // the pointer for the i-th line; this line uses p[0], p[1], ..., p[k-1];
    // the remaining elements of this line will never be used by a correct program
    a[i] = p - (n - 1) + i;
    // advance p
    p += k;
  }
}
```

1.5 **a)** Calcule o valor a dar à variável `s` de modo a que seja alocado o número exato de elementos da matriz.

1.5 **b)** Num acesso á matriz usando `a[i][j]`, qual é a gama de valores válidos para `j`?

1.0 **c)** Qual é a forma da matriz?

1.5 **7:** Dê um exemplo de uma função que tenha uma complexidade computacional de $\Theta(n^2)$.