Teste de recurso de **Algoritmos e Estruturas de Dados**

29 de janeiro de 2018 **Duração: 3 hora**

Justifique todas as suas respostas.

Nas perguntas sobre árvores binárias, cada nó da árvore usa a seguinte estrutura de dados:

```
typedef struct tree_node
{
  struct tree_node *left;   // pointer to the left branch (a sub-tree)
  struct tree_node *right;  // pointer to the right branch (a sub-tree)
  struct tree_node *parent; // pointer to the parent node (NULL for the root node)
  int data;                 // the data (only one node can have a given data value)
}
tree_node;
```

Nome:

N. Mec.:

Fórmulas:

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

1.5 **1:** No seguinte código, para que valores da variável `i` é avaliada a função `g(x)` e que valores de `i` são impressos?

```
#include <stdio.h>

int f(int x) { return 2 * x - 5; }
int g(int x) { return x * x - 50; }

int main(void)
{
  for(int i = -10;i <= 10;i++)
    if( (f(i) > 0) || (g(i) > 0) )
      printf("%d\n",i);
  return 0;
}
```

1.0 **2:** No seguinte código,

```
int a[100],*b = &a[17];
```

qual é o índice do elemento do array `a` que é referenciado por `b[+14]`?

1.0 **3:** Ordene as seguintes funções por ordem crescente de ritmo de crescimento.

Número da função	função
1	$\frac{n!}{n^{100}} - 1$
2	$n^2 \log n + \sqrt{n}$
3	$1.1^n + 17 + n^3$
4	$23 + \frac{\log^2 n}{n}$
5	$n^2 + \frac{10^{100}}{n}$

1.5 **4:** Qual é o valor devolvido pela 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 **5:** Dê um exemplo de uma função que tenha uma complexidade computacional de $\boldsymbol{\Theta(n^3)}$.

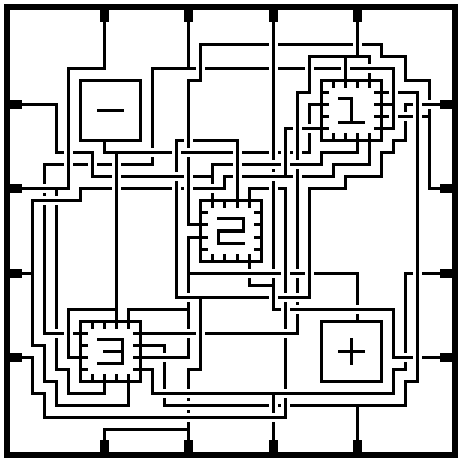
2.0 **6:** Escreva uma função recursiva **eficiente** que, dada a raíz de uma árvore binária **ordenada**, dois valores `lo` e `hi`, conta o número de nós da árvore que armazenam valores entre `lo` e `hi`. Qual é a complexidade computacional da sua função?

2.0 **7:** Explique como está organizada a informação num *min-heap*. Ilustre a sua exposição inserindo os números, por esta ordem, num *min-heap* inicialmente vazio: **7**, **3**, **9**, **1**, **2**.

1.5 **8:** Compare dois algoritmos de ordenação à sua escolha no que diz respeito a i) complexidade computacional, ii) melhor caso, iii) pior caso.

2.0 **9:** Explique como funciona uma *hash table*. Indique as vantagens e desvantagens das implementações de *hash tables* usando *open-addressing* e *chaining*.

2.0 **10:** Considere o labirinto recursivo da figura seguinte.



Neste labirinto os quadrados com os números **1**, **2** e **3** são cópias do quadrado todo, ou seja, dentro de cada um desses quadrados está o quadrado todo (daí o labirinto ser recursivo). Pretende-se ir do quadrado $+$ para o quadrado $-$, **no mesmo nível**, isto é, a ordem por que se sai dos quadrados **1**, **2** e **3** tem de ser a ordem inversa por que se entra nesses mesmos quadrados. Por exemplo, uma ordem possível de entradas e saídas é: entra em 1, entra em 3, entra em 2, sai de 2, entra em 1, sai de 1, sai de 3, sai de 1. (No nível exterior não se pode sair do quadrado.)

Descreva, **por palavras**, um algoritmo capaz de resolver o problema. (Não é preciso fazer o programa!) Pode usar *depth-first search* para resolver este problema?

2.0 **11:** Considere um rectângulo de largura $\boldsymbol{w}$ e altura $\boldsymbol{h}$. Pretende-se contar o número de maneiras de ir do canto inferior esquerdo (coordenadas $\boldsymbol{x = 1}$ e $\boldsymbol{y = 1}$) para o canto superior direito (coordenadas $\boldsymbol{x = w}$ e $\boldsymbol{y = h}$) com um custo que não ultrapassa **10**, fazendo movimentos dos seguintes tipos:

- movimento para a direita ($\boldsymbol{\Delta x = 1}$, $\boldsymbol{\Delta y = 0}$), custo **0**;
- movimento para cima ($\boldsymbol{\Delta x = 0}$, $\boldsymbol{\Delta y = 1}$), custo **0**;
- movimento para a direita e para cima ($\boldsymbol{\Delta x = 1}$, $\boldsymbol{\Delta y = 1}$), custo **0**;
- movimento para a esquerda ($\boldsymbol{\Delta x = -1}$, $\boldsymbol{\Delta y = 0}$), custo **1**;
- movimento para baixo ($\boldsymbol{\Delta x = 0}$, $\boldsymbol{\Delta y = -1}$), custo **1**; e
- salto de cavalo ($\boldsymbol{\Delta x = 2}$, $\boldsymbol{\Delta y = 1}$), custo **2**.

Escreva código `C` capaz de resolver o problema (use o tipo de dados `long` para armazenar números de movimentos.) Que estratégia algoritmica usou, e qual a sua complexidade computacional?

2.0 **12:** Um grafo tem **5** vértices. Os vértices desse grafo estão numerados de **1** a **5**; entre o vértice número $\boldsymbol{i}$ e o vértice número $\boldsymbol{j}$ só existe uma aresta se apenas um desses dois números for um número primo.

- Desenhe o gráfico.
- Represente o grafo pela sua matriz de adjacência.
- Represente o grafo por listas de adjacência.
- Quantas maneiras diferentes existem para se ir do vértice **1** para o vértice **2** sem nunca passar pela mesma aresta duas vezes?