Questões de introdução à programação

Linguagens de programação aceitas: **C, C++, Fortran, Java e Pascal**

1. Escreva um código que leia do usuário um número inteiro positivo N e calcule:

- $\sum_{a=1}^{N}\sum_{b=1}^{N} f(a,b)$
- $\max_{1 \le a,b \le N} f(a,b)$
- $\min_{1 \le a,b \le N} f(a,b)$

onde $f(a,b) = 3ab - 2a + 5b + 10$***, e mostre esses valores para o usuário.***

2. Escreva uma **função**, denominada **reorder**, que recebe um array unidimensional (vetor) de números inteiros e reordena os elementos *nesse mesmo array* de forma que todas as seguintes condições sejam satisfeitas na ordem dos valores no retorno da função:

- Todos os valores originalmente no array deve ser mantidos e nenhum novo valor acrescentado (inclusive respeitando as multiplicidades originais).
- Todos os valores ímpares devem preceder todos os valores pares.
- Os valores ímpares devem estar ordenados em ordem crescente.
- Os valores pares devem estar ordenados em ordem descrescente.

Por exemplo, se o array tem os seguintes valores ao entrar na função:

```
1 -2 7 0 8 -5  0 11
```

Então ele deve apresentar na saída da função os valores na seguinte ordem:

```
-5 1 7 11 8 0 0 -2
```

Inglês:

1. Write a program that reads from the user a positive integer value N, computes the following:

- $\sum_{a=1}^{N}\sum_{b=1}^{N} f(a,b)$
- $\max_{1 \le a,b \le N} f(a,b)$
- $\min_{1 \le a,b \le N} f(a,b)$

where $f(a,b) = 3ab - 2a + 5b + 10$***, and shows these values to the user.***

2. Write a **function**, named **reorder**, that receives an 1D array (vector) of integer numbers and reorders them *in the same array* such that the final ordering on return from the function satisfies all of the following conditions:

- All the values in the input array, with their multiplicity, are preserved, and no new values are inserted.
- All odd values must come before all even values.
- Odd values must be ordered in increasing order.
- Even values must be ordered in decreasing order.

For example, if the array has the following values (in order) on function entry:

1 −2 7 0 8 −5 0 11

on function exit the values must be ordered as

−5 1 7 11 8 0 0 −2

Respostas:

1-
(Uma possível solução em C++):

```
#include <iostream>
#include <limits>

using namespace std;

int main(int, char *[])
{
    int N;
    cout << "Value of N: ";
    cin >> N;

    int sum = 0;
    int max = numeric_limits<int>::lowest();
    int min = numeric_limits<int>::max();
    for (int a = 1; a <= N; ++a) {
        for (int b = 1; b <= N; ++b) {
            auto f_ab = 3*a*b - 2*a + 5*b + 10;
            sum += f_ab;
            max = f_ab > max ? f_ab : max;
            min = f_ab < min ? f_ab : min;
        }

    }

    cout << "Sum: " << sum << endl;
    cout << "Max: " << max << endl;
    cout << "Min: " << min << endl;
}
```

2-
(Uma possível solução em C++)

```
#include <vector>
#include <algorithm>
#include <functional>

void reorder(std::vector<int> &v)
{
    int N = v.size();
    int odd_end = 0;
    int even_start = N;
    while (odd_end != even_start) {
```

```
        if (v[odd_end] % 2 != 0) {
            ++odd_end;
        }
        else if (v[even_start-1] % 2 == 0) {
            --even_start;
        }
        else {
            std::swap(v[odd_end], v[even_start - 1]);
            ++odd_end;
            --even_start;
        }
    }
    std::sort(v.begin(), v.begin() + odd_end);
    std::sort(v.begin() + odd_end, v.end(), std::greater<int>());
}
```

<u>QUESTÕES MÉTODOS NUMÉRICOS:</u>

1. Apresente a fórmula básica (interativa) do método de busca de raízes conhecido como Newton-Raphson aplicado para a busca de uma raiz de uma função f(x).

1. Present the basic formula (interactive) underlying the root finding algorithm known as Newton-Raphson as applied for searching for a root of a function f(x).

Resposta 1:
x_(i+1) = x_i = f(x_i)/f'(x_i)

<u>Questões Estrutura de Dados</u>

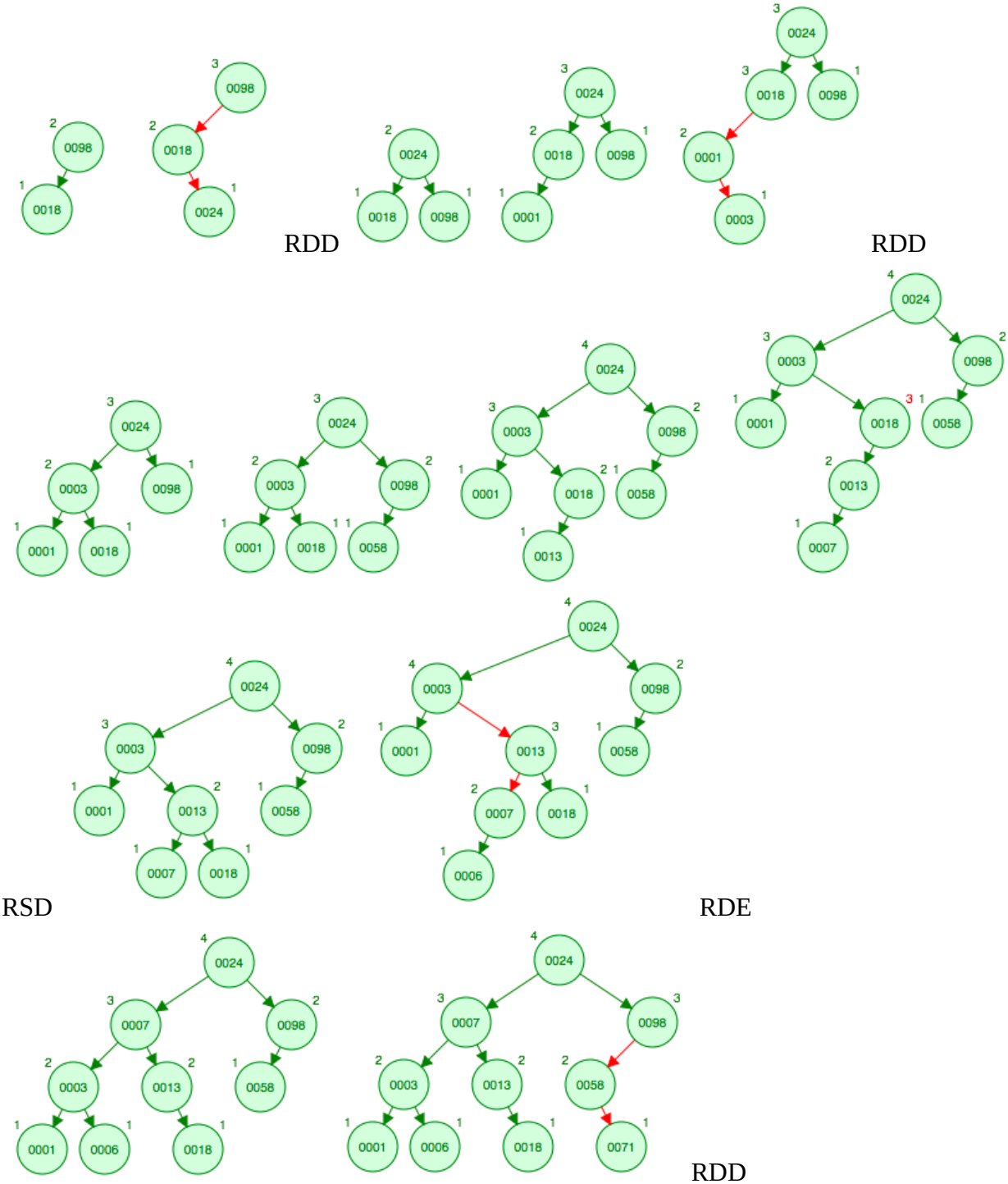
Português:
1 - Transforme o vetor **a** em uma árvore AVL, seguindo a ordem dos elementos do vetor. Mostre os passos para cada elemento inserido:
a = (98, 18, 24, 1, 3, 58, 13, 7, 6, 71, 21, 19, 8)

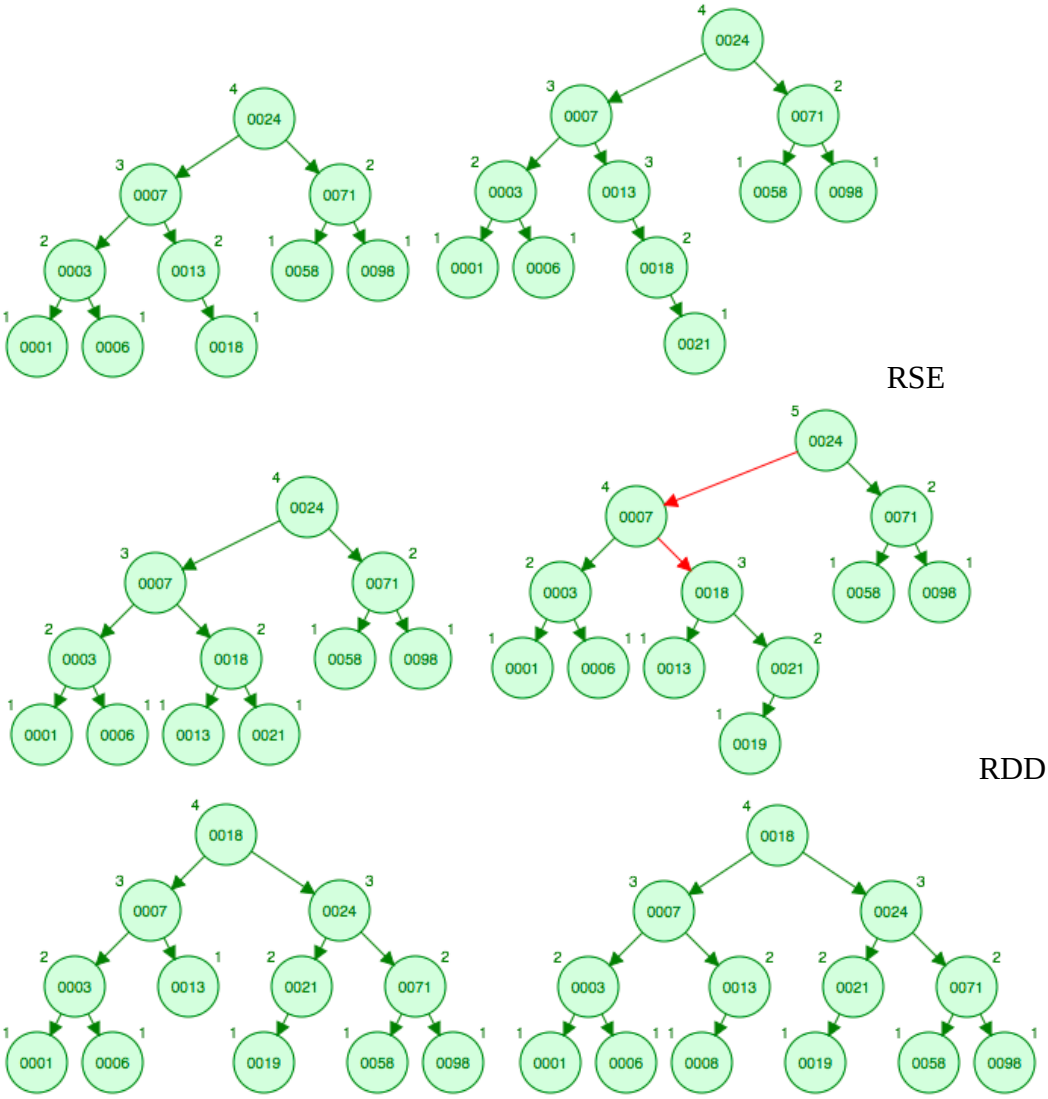
2 – Apresente uma função que retorne o grau máximo de um grafo, quantos vértices possuem grau máximo e quais são. O grafo deve ser representado por sua matriz de adjacência.

Inglês:
1 - Transform the vector **a** into an AVL tree, following the order of vector elements. Show the steps for each inserted element:
a = (98, 18, 24, 1, 3, 58, 13, 7, 6, 71, 21, 19, 8)

2 - Write a function that returns the maximum degree of a graph, how many vertices have a maximum degree, and what are they. The graph must by represented by its adjacency matrix.

Resposta 1:





Resposta 2:

Algoritmo genérico (pseudo algoritmo, sem linguagem específicada e com traços de C), detalhes mudam de linguagem para linguagem. Nesta resposta é apresentado a forma geral, sem as especificações para lidar com vetores e matrizes ou com funções auxiliares.

```
int[] exercicio (int[][] matrizAdj){
        int n = tamanho_de_matrizAdj;    // deve ser definida a forma de encontrar o tamanho
        int maxG = 0;
        int grau;
        int[] maxE;
        int c = 0;
        int[3] saida;   // a saida precisa de uma estrutura de dados para retornar um vetor no
                        // elemento 2. Uma alternativa seria passar o endereço do parâmetro na
                        // função
        int j,k;

        for j = 1:n
        {
                grau = 0;
```

```
		for k = 1:n
		{
			if (grafo[j,k]==1)
			{
				grau = grau + 1;
			}
		}
		if (maxG == grau) {
			c = c + 1;
			maxE[c] = j;
		}
		if (maxG < grau) {
			maxG = grau(j);
			maxE[1] = j;
			c = 1;
		}
	}
	saida[0] = magG
	saida[1] = c
	saida[2] = maxE
	retorna(saida);
}
```