**Instituto Federal de Educação, Ciência e Tecnologia**

Câmpus Campinas
**ED1 – Estruturas de Dados I**

**Professor:** Samuel Martins (samuel.martins@ifsp.edu.br)

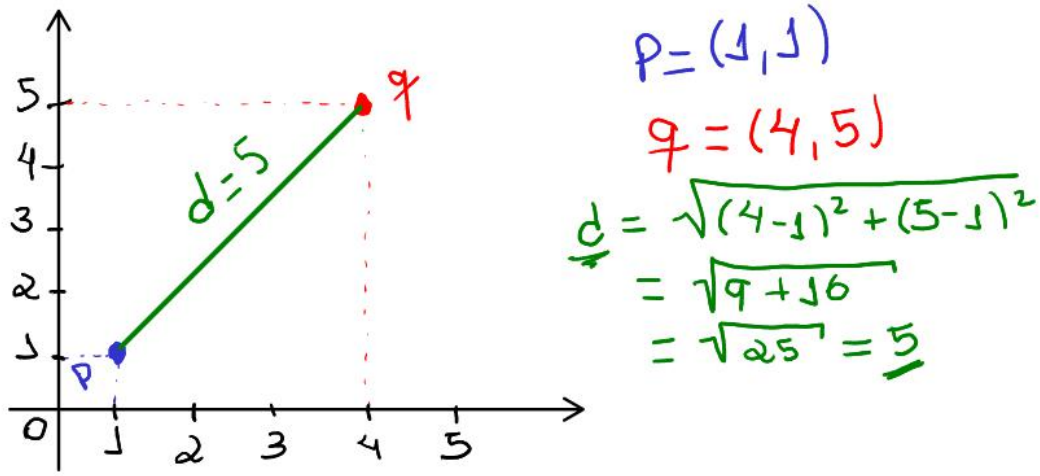
# Distância entre Pontos

## 1. Descrição

Reginaldo é um estudo muito aplicado que adora matemática. Em seus estudos de geometria, ele aprendeu como calcular a **distância Euclideana** entre dois pontos no espaço.

Dado 2 pontos **p** e **q** $\in \mathbb{R}^n$, ou seja, $p = (p_1, p_2, \cdots, p_n)$ e $p = (q_1, q_2, \cdots, q_n)$, a **distância Euclideana** entre p e q é dada pela fórmula: $d = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$.

Exemplo:
p = (1, 1), q = (4, 5), n = 2 (dimensões)




Escreva um programa que, dado dois pontos com n dimensões, compute a distância Euclideana dos pontos.

## 2. Especificação da Entrada e Saída

### Entrada

A primeira linha da entrada corresponde a um **número inteiro N** ($N \geqslant 1$), que indica o número de dimensões (coordenados) dos pontos.

As próximas **n linhas** contém as coordenadas do primeiro ponto.

As **n linhas** subsequentes contém as coordenadas do segundo ponto.

Todas as coordenadas são representadas por números decimais.

### Saída

Seu programa deve escrever na saída *uma única linha* contendo a frase: '*A distancia Euclideana entre os pontos eh: X*', onde X é a distância Euclideana computada com precisão de **duas casas decimais**. Utilize a flag **%.2f** no printf.

**Exemplos**

n = 2, p = (1, 1), q = (4, 5)

| Entrada | Saída |
|---|---|
| 2<br>1<br>1<br>4<br>5 | A distancia Euclideana entre os pontos eh: 5.00 |

n = 3, p = (1, 2, 3), q = (10, 20, 30)

| Entrada | Saída |
|---|---|
| 3<br>1<br>2<br>3<br>10<br>20<br>30 | A distancia Euclideana entre os pontos eh: 33.67 |

## 3. Observações Gerais

- Utilize a flag **"%.2f"** para imprimir um float com precisão de casais decimais;
- **Apenas um integrante** da dupla deverá submeter o código;
- Caso ambos submetam, será o considerado o código da última submissão;
- A nota é dada pelo **número de casos de teste acertados**;
- Codigos com **erros de compilação e execução**, tais como Segmentation Fault, **serão considerados errados**;
- Utilize ***return 0;*** na main de seu programa;
- Qualquer tentativa de fraude, plagio e afins, correspondera em **nota ZERO** para os envolvidos;
- **Códigos ilegíveis serão considerados errados**. A legibilidade é obtida com identação correta e coerente, bons nomes de variáveis e funções, bem como **boa subdivisão do código** em funções auxiliares;

## 4. Dicas

- Para **compilar** seu código no terminal:
  - ***gcc lab.c -o lab***
- **-o** significa *output*. Ele é responsável por gerar o binário do seu programa para execução. É **OBRIGATÓRIO** que o arquivo tenha a função **main;**
- Logo, o que você está dizendo é: "*compile o código **lab.c** com o compilador **gcc**, gerando o executável (saída) **lab**";*

- Para **executar** seu programa:
  - ***./lab***

- Você pode baixar os arquivos de casos de teste do run.codes e executá-los manualmente:
  - ***./lab < 01.in***
- A diretiva **<** redireciona o conteúdo do arquivo *01.in* para o terminal, cujas entradas/dados serão lidas pelo ***scanf***;
- Você pode ainda redirecionar a *saída* impressa no terminal para um arquivo:
  - ***./lab < 01.in > 01.res***

- Por fim, você poder comparar sua reposta com o gabarito (resultado do caso de teste), fazendo
  - ***diff 01.res 01.out***
  - onde *01.out* é a saída esperada para a entrada *01.in*