=Universidade Tecnológica Federal do Paraná - Campus Campo Mourão (UTFPR-CM)
Unidade curricular: Introdução à Programação (BCC1001) do Bacharelado em Ciência da Computação
Tipo de avaliação: Prova 1 - Teórica **Data:** 04/10/2023

Estudante: _____

1. Considere as seguintes afirmações e classifique-as entre verdadeiras e falsas. Caso a afirmação seja falsa, explique o que está errado nela (se possível, use exemplos para complementar a explicação) **(Valor: 2,0 quanto à classificação e acréscimo de 0,25 por cada justificativa correta.)**

(F) a) Algoritmo é a descrição de uma conjunto aleatório de passos que devem ser seguidos para a realização de uma tarefa.

(V) b) Um programa de computador é uma representação formal de um algoritmo em uma linguagem de programação, com a definição de comandos e estruturas de dados para processar dados obtidos de dispositivos de entrada ou em memória, com eventual apresentação dos resultados da computação para dispositivos de saída ou armazenamento de resultados em memória.

(F) c) Existem várias maneiras de implementar um algoritmo em uma linguagem de programação. No entanto, existe apenas um algoritmo possível para resolver um problema específico.

(V) d) Para criar um programa de computador, é recomendável usar um ambiente de programação específico. Para Python, um desses ambientes é o Thonny.

(V) e) De modo geral, o computador executa um programa de forma muito rápida. Uma forma de controlar a velocidade de execução de um programa é utilizar o modo de depuração oferecido pelos ambientes de programação.

Justificativas:

a) Um algoritmo define uma sequência de passos. Não é um conjunto (sem ordem entre os passos) ou aleatório (com uma ordem qualquer). Por exemplo, para a corrida de saco, entrar no saco, flexionar as pernas e pular precisa ser feito nessa ordem: qualque outra ordem resultará em uma solução inválida para o problema.

c) Para um problema específico podem existir vários algoritmos. Por exemplo, para a ordenação, temos vários algoritmos (Bubblesort, Insertion sort, Quicksort) que resolvem o mesmo problema.

2. Considere o seguinte programa e responda os itens desta questão: (**Valor: 2,4**)

Linha	Programa
1	`a = float(input())`
2	`b = (3 + ((17 // 3) * 4**2) % 2)`
3	`c = -b / 2`
4	`b /= b`

a) Quais são os valores das variáveis 'b' e 'c' após a execução das linhas 1, 2 e 3?
A variável 'b' terá o valor 3 e a variável 'c' terá o valor -1,5.

b) O que faz o comando da linha 4?
O comando atualizará o valor da variável 'b' com o resultado d a divisão entre ele mesmo (b) e o operando do lado direito da operação /= do comando (que no caso também é a variável 'b'). Ao final disso, o valor de 'b' será 1,0.

c) Quais são as variáveis definidas e seu tipo de dado durante a execução do programa? Informe isso considerando o resultado de execução de cada linha do programa.

```
Linha 1: a, de tipo float, com o valor de float convertido a partir do texto informado pelo usuário.

Linha 2: a, de tipo float, conforme mencionado na linha 1; b = 3, de tipo integer.

Linha 3: a, de tipo float, conforme mencionado na linha 1; b = 3, de tipo integer; c = -1,5, de tipo float

Linha 4: a, de tipo float, conforme mencionado na linha 1; b = 1,0, de tipo float; c = -1,5, de tipo float
```

3. Considere duas cidades, sendo que a primeira possui uma população de 9.000 enquanto que a segunda possui 6.000. A taxa de crescimento delas é bem peculiar: a primeira cidade cresce a uma taxa anual de 1% enquanto que a segunda cresce a 20%. Logo, eventualmente a segunda cidade ultrapassará a população da primeira. Agora responda: **(Valor: 2,4)**

a) Quais seriam as **variáveis** necessárias para representar computacionalmente os principais elementos do problema acima?

As variáveis necessárias são:

população_a, de tipo integer

população_b, de tipo integer

taxa_a, de tipo float

taxa_b, de tipo float

b) **Esboce o algoritmo** (ou, se preferir, o programa em Python) que calcula a população de cada cidade após a quantidade de anos necessária para que a segunda cidade ultrapasse a primeira cidade, *compara as populações e informe se a segunda cidade possui mais habitantes que a primeira cidade (parte em itálico é opcional).*

Algoritmo:

Definir tamanho da população da primeira cidade

Definir taxa de crescimento da população da primeira cidade

Definir tamanho da população da segunda cidade

Definir taxa de crescimento da população da segunda cidade

Inicializar com zero um contador (variável integer) para os anos

Atualizar o tamanho da primeira cidade conforme a taxa de crescimento dela para um ano

Atualizar o tamanho da segunda cidade conforme a taxa de crescimento dela para um ano

Incrementar o contador de anos

Programa:

```
população_a = 9000
população_b = 6000
taxa_a = 0.01
taxa_b = 0.20
anos = 0
população_a += população_a * taxa_a
população_b += população_b * taxa_b
anos += 1
print("%d - A: %d \t B: %d" % (anos, população_a, população_b))
```

```
população_a += população_a * taxa_a
população_b += população_b * taxa_b
anos += 1
print("%d - A: %d \t B: %d" % (anos, população_a, população_b))
população_a += população_a * taxa_a
população_b += população_b * taxa_b
anos += 1
print("%d - A: %d \t B: %d" % (anos, população_a, população_b))
```

c) No algoritmo feito no item (b), provavelmente existem trechos (ou linhas de código) que se repetem. Como você substituiria essas repetições pela definição de uma função e por chamadas à essa função em seu algoritmo?

Programa com variáveis globais:

```
def atualizar_populações():
    global população_a
    global população_b
    global anos
    população_a += população_a * taxa_a
    população_b += população_b * taxa_b
    anos += 1
    print("%d - A: %d \t B: %d" % (anos, população_a, população_b))

população_a = 9000
população_b = 6000
taxa_a = 0.01
taxa_b = 0.20
anos = 0
atualizar_populações()
atualizar_populações()
atualizar_populações()
```

Programa com passagem de parâmetros:

```
def atualizar_população(população, taxa):
    população += população * taxa
    return população

população_a = 9000
população_b = 6000
taxa_a = 0.01
taxa_b = 0.20
anos = 0
população_a = atualizar_população(população_a, taxa_a)
população_b = atualizar_população(população_b, taxa_b)
anos += 1
print("%d - A: %d \t B: %d" % (anos, população_a, população_b))
população_a = atualizar_população(população_a, taxa_a)
população_b = atualizar_população(população_b, taxa_b)
anos += 1
print("%d - A: %d \t B: %d" % (anos, população_a, população_b))
população_a = atualizar_população(população_a, taxa_a)
população_b = atualizar_população(população_b, taxa_b)
anos += 1
print("%d - A: %d \t B: %d" % (anos, população_a, população_b))
```

4. A distância entre dois pontos pode ser calculada considerando o teorema de Pitágoras, no que denominamos como distância euclidiana. Cada ponto possui uma posição (x, y) no plano e a distância seria a medida do segmento de reta que liga os dois pontos. Para o ponto inicial A teríamos então a posição (x_A, y_A) e para o ponto final B teríamos (x_B, y_B). A distância d seria calculada pela raiz quadrada da soma dos quadrados das diferenças entre x_B e x_A e entre y_B e y_A, ou seja, $d = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$. No quadro a seguir é apresentado o código de um programa que calcula e mostra a distância entre os pontos A e B e entre B e A, informados pelo usuário, representados pelas variáveis `a_x`, `a_y`, `b_x` e `b_y`. Esse programa define e faz uso da função `distância_euclidiana` para calcular a distância. No entanto, o código está fora de ordem (embora a indentação esteja correta). Corrija a ordem das linhas do programa e mostre o resultado da execução do programa para os pontos A = (1, 2) e B = (3, 5). **(Valor: 2,7)**

```
print("A distância entre B e A é de %.2f" % distancia_b_a)
import math
a_y = int(input("Coordenada y do ponto A: "))
    delta_x = b_x - a_x
distancia_a_b = distância_euclidiana(a_x, a_y, b_x, b_y)
    distância = math.sqrt(delta_x**2 + delta_y**2)
print("A distância entre A e B é de %.2f" % distancia_a_b)
    delta_y = b_y - a_y
distancia_b_a = distância_euclidiana(b_x, b_y, a_x, a_y)
a_x = int(input("Coordenada x do ponto A: "))
b_x = int(input("Coordenada x do ponto B: "))
def distância_euclidiana(a_x, a_y, b_x, b_y):
b_y = int(input("Coordenada y do ponto B: "))
    return distância
```

Uma possível resposta:

```
import math
def distância_euclidiana(a_x, a_y, b_x, b_y):
    delta_x = b_x - a_x
    delta_y = b_y - a_y
    distância = math.sqrt(delta_x**2 + delta_y**2)
    return distância
a_x = int(input("Coordenada x do ponto A: "))
a_y = int(input("Coordenada y do ponto A: "))
b_x = int(input("Coordenada x do ponto B: "))
b_y = int(input("Coordenada y do ponto B: "))
distancia_a_b = distância_euclidiana(a_x, a_y, b_x, b_y)
distancia_b_a = distância_euclidiana(b_x, b_y, a_x, a_y)
print("A distância entre A e B é de %.2f" % distancia_a_b)
print("A distância entre B e A é de %.2f" % distancia_b_a)
```

Este exercício pode permitir inverter as linhas 4 e 8 e colocar em qualquer ordem as linhas do conjunto 10, 3, 11 e 13 (que podem estar no bloco de código antes ou após a definição da função) e do conjunto de linhas 5 e 9.

Resultados:

```
A distância entre A e B é de 3.61
A distância entre B e A é de 3.61
```