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 **Data:** 02/05/2024

Estudante: _____

1. Considere as seguintes afirmações. Identifique as incorretas e explique a razão delas serem incorretas. Para as afirmações corretas, dê um exemplo referente à afirmação. **(Valor: 2,0)**

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

Afirmação correta. Por exemplo, o algoritmo para verificar se um número é par pode ser definido com a seguinte sequência de passos:

1. Obtenha o número a ser verificado.
2. Realize a divisão inteira desse número com o número 2.
3. Caso o resto da divisão seja zero, o número é par. Caso contrário, o número é ímpar.

( ) 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.

Afirmação correta. Considere o algoritmo apresentado no item anterior. O seguinte programa pode ser definido na linguagem de programação Python:

```
número = int(input())
resto = número % 2
if resto == 0:
      print("Número é par")
else:
      print("Número é ímpar")
```

Neste programa, temos comandos de entrada e saída (chamada à função *input* para obter o cado do usuário), tratamento de dados obtidos na entrada chamada à função *int* para converter o dado informado pelo usuário em um número inteiro), com armazenamento do resultado em variável; resolução de expressão aritmética (*número % 2*) e armazenamento do resultado dessa resolução em uma variável; comando de seleção com relação ao valor desta última variável e o valor zero, para escolher o fluxo de execução entre informar que um número é par ou ímpar, mostrando o resultado na saída padrão do computador.

(X) 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.

Esta afirmação está incorreta. De fato, existem várias maneiras de implementar um algoritmo em uma linguagem de programação. No entanto, aí está o erro da afirmação, existem vários algoritmos para resolver a maioria dos problemas computacionais. Por exemplo, no primeiro dia de aula, vimos exemplos de soluções para o problema de ordenação. Além de observar implementações em várias linguagens do algoritmo de ordenação *Bubblesort*, também foram mostrados outros algoritmos de ordenação: *Selection sort* e *Quicksort*.

(X) d) Uma variável representa um conceito relacionado ao problema ou a sua solução. Toda variável em Python possui um tipo e um valor daquele tipo. Os principais tipos disponíveis em Python são: int, decimal,

float e texto.

Esta afirmação está incorreta. O erro reside nos tipos de dados disponíveis em Python, linguagem que não contém o tipo *decimal*. Observe que existe o módulo *decimal* que provê esse tipo de dado, mas esse módulo não é carregado automaticamente (embora faça parte da biblioteca padrão do Python).

( ) e) Um comando de seleção permite alterar o fluxo de execução de um programa a partir da avaliação de uma expressão com resultado Booleano ao se utilizar um comando if-elif-else ou a partir da avaliação do valor de uma variável e o casamento com um padrão especificado ao se utilizar o comando match-case.

Esta afirmação está correta. No segundo item desta questão temos um exemplo de comando if-else que avalia o valor da variável *resto* em relação ao valor 0 quando à igualdade. Caso optássemos por usar o comando match-case, teríamos o seguinte código:

```
número = int(input())
resto = número % 2
match resto:
      case 0:
            print("Número é par")
      case 1:
            print("Número é ímpar")
```

2. Considere o seguinte programa e responda os itens desta questão:

Linha	**Programa**
1	`a = input()`
2	`a = int(a)`
3	`b = 2**3 + (3 + ((15 // 3) * 4) % 2)`
4	`c = -b / 2`
5	`a *= b`

a) Quais são os valores das variáveis 'b' e 'c' após a execução do programa? **(Valor: 0,5)**
Nesse caso, os valores das variáveis b e c seriam, respectivamente, 11 e -5.5.

b) O que faz o comando da linha 5? Informe também o resultado (incluindo valores e tipos de dados envolvidos e resultantes) deste comando. **(Valor: 0,5)**
O comando da linha 5 define a variável *a* a partir do valor atual da variável *a* multiplicado pelo valor da variável *b*. Considere que o valor informado pelo usuário na Linha 1 foi '3'. Neste caso, temos que o valor atual de *a* é 3 e que o novo valor de *a* será 33.

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 (e informe isso linha a linha). **(Valor: 1,0)**
Linha 1: definição da variável *a*, de tipo *string* e valor *'3'*.
Linha 2: definição da variável *a*, de tipo *int* e valor *3*.
Linha 3: definição da variável *b*, de tipo *int* e valor 11. Mantém-se a variável *a* conforme definida na Linha 2.
Linha 4: definição da variável *c*, de tipo *float* e valor *5.5*. Mantêm-se as variáveis *a* e *b* conforme definidas nas Linhas 2 e 3, respectivamente.
Linha 5: definição da variável *a*, de tipo *int* e valor 33. Mantêm-se as variáveis *b* e *c* conforme definidas nas Linhas 3 e 4, respectivamente.

3. Considere duas cidades, sendo que a primeira possui uma população de 5.000 enquanto que a segunda possui 6.000. A taxa de crescimento delas é bem peculiar: a primeira cidade cresce a uma taxa anual de 10% enquanto que a segunda cresce a 5%. Logo, eventualmente a primeira cidade ultrapassará a população da segunda. Agora responda: **(Valor: 3,0)**

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

As variáveis necessárias para representar os principais elementos do problema são: *populaçãoA* e *populaçãoB*, do tipo *int* e que representam, respectivamente, a população da primeira e da segunda cidade; *taxaA* e *taxaB*, do tipo *float* e que representam, respectivamente, a taxa anual de crescimento da primeira e da segunda cidade.

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 primeira cidade ultrapasse a segunda cidade, *compara as populações e informe se a segunda cidade possui mais habitantes que a primeira cidade (parte em itálico é opcional).*

```
populaçãoA = 5000
populaçãoB = 6000
taxaA = 0.1
taxaB = 0.05
anos = 0
se populaçãoA > populaçãoB, mostrar "Primeira cidade é maior do que a segunda cidade"

populaçãoA = populaçãoA + (populaçãoA * taxaA)
populaçãoB = populaçãoB + (populaçãoB * taxaB)
anos = anos + 1
se populaçãoA > populaçãoB, mostrar "Primeira cidade é maior do que a segunda cidade"

populaçãoA = populaçãoA + (populaçãoA * taxaA)
populaçãoB = populaçãoB + (populaçãoB * taxaB)
anos = anos + 1
se populaçãoA > populaçãoB, mostrar "Primeira cidade é maior do que a segunda cidade")


populaçãoA = populaçãoA + (populaçãoA * taxaA)
populaçãoB = populaçãoB + (populaçãoB * taxaB)
anos = anos + 1
se populaçãoA > populaçãoB, mostrar "Primeira cidade é maior do que a segunda cidade")

populaçãoA = populaçãoA + (populaçãoA * taxaA)
populaçãoB = populaçãoB + (populaçãoB * taxaB)
anos = anos + 1
se populaçãoA > populaçãoB, mostrar "Primeira cidade é maior do que a segunda cidade"
```

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? Esboce o algoritmo ou o programa com essas alterações.

No caso do algoritmo definido no item anterior, a atualização das populações e do ano e a comparação entre o tamanho das cidade é realizado quatro e cinco vezes, respectivamente. Poderíamos criar uma função para cada uma dessas responsabilidades: uma função para conferir a população e mostrar a mensagem conforme o tamnho e outra função para atualizar as populações e anos. Uma forma de fazê-lo seria como o mostrado abaixo (no caso usando variáveis globais).

```
def conferir_população(populaçãoA, populaçãoB):
    if populaçãoA > populaçãoB:
        print("Após %d anos a primeira cidade terá população maior que a segunda" % anos)
        return True
    else:
        return False
```

```
def atualizar_população(taxaA, taxaB):
    global populaçãoA, populaçãoB, anos
    populaçãoA = populaçãoA + (populaçãoA * taxaA)
    populaçãoB = populaçãoB + (populaçãoB * taxaB)
    anos = anos + 1


populaçãoA = 5000
populaçãoB = 6000
taxaA = 0.1
taxaB = 0.05
anos = 0

if not conferir_população(populaçãoA, populaçãoB):
    atualizar_população(taxaA, taxaB)

if not conferir_população(populaçãoA, populaçãoB):
    atualizar_população(taxaA, taxaB)

if not conferir_população(populaçãoA, populaçãoB):
    atualizar_população(taxaA, taxaB)

if not conferir_população(populaçãoA, populaçãoB):
    atualizar_população(taxaA, taxaB)

if not conferir_população(populaçãoA, populaçãoB):
    atualizar_população(taxaA, taxaB)
```

Também poderíamos utilizar uma única função para conferir e atualizar a população, sem usar variáveis globais:

```
def conferir_atualizar_população(populaçãoA, taxaA, populaçãoB, taxaB, anos):
    populaçãoA = populaçãoA * (1 + taxaA)**anos
    populaçãoB = populaçãoB * (1 + taxaB)**anos
    if populaçãoA > populaçãoB:
        print("Após %d anos a primeira cidade terá população maior que a segunda" % anos)
        return True
    else:
        return False

populaçãoA = 5000
populaçãoB = 6000
taxaA = 0.1
taxaB = 0.05
anos = 0

if not conferir_atualizar_população(populaçãoA, taxaA, populaçãoB, taxaB, anos):
    anos = anos + 1
if not conferir_atualizar_população(populaçãoA, taxaA, populaçãoB, taxaB, anos):
    anos = anos + 1
if not conferir_atualizar_população(populaçãoA, taxaA, populaçãoB, taxaB, anos):
    anos = anos + 1
if not conferir_atualizar_população(populaçãoA, taxaA, populaçãoB, taxaB, anos):
    anos = anos + 1
if not conferir_atualizar_população(populaçãoA, taxaA, populaçãoB, taxaB, anos):
    anos = anos + 1
```

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 deveria calcular e mostrar a distância entre os pontos A e B, 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 com alguns erros. Identifique os erros e os corrija. Após isso, mostre o resultado (aproximado) da execução do programa para os pontos A = (3, 5) e B = (5, 9). **(Valor: 3,0)**

```
1   import everything from math
2
3   a_x = input(int("Coordenada x do ponto A: "))
4   a_y = input(int("Coordenada y do ponto A: "))
5   a_x = input(int("Coordenada x do ponto B: "))
6   a_y = input(int("Coordenada y do ponto B: "))
7
8   distancia_a_b = distância_euclidiana()
9   print("A distância entre A e B é de .2f" % distancia_a_b)
10
11  def distância_euclidiana(1a, 2a, 1b, 2b):
12      delta_x = 1b - 1a
13      delta_y = 2b - 2a
14      distância = math.sqrt(delta_x**2 + delta_y*2)
```

O código possui os seguintes erros:

- Linha 1: A importação do módulo *math* foi realizada incorretamente. Para este programa, basta importar o módulo, sem especificar elementos específicos dele: *import math*
- Linhas 3, 4, 5 e 6: Todas essas linhas possuem o mesmo tipo de erro: foi invertida a ordem de chamada das funções. O correto seria chamar primeiro a função *input* e depois a *int*: `int(input(""))`
- Linha 5: A variável que deveria ser definida é *b_x* ao invés de *a_x*.
- Linha 6: A variável que deveria ser definida é *b_y* ao invés de *a_y*.
- Linha 8: A chamada à função *distância_euclidiana* não apresenta os quatro argumentos especificados na assinatura da função: distancia_a_b = distância_euclidiana(a_x, a_y, b_x, b_y)
- Linha 9: Na parametrização da string formatada, faltou adicionar o % antes do .2f: print("A distância entre A e B é de %.2f" % distancia_a_b)
- Linha 11: O nome dos parâmetros começa com um número, mas um nome não pode começar com um número em Python. Uma solução seria inverter, ficando: def distância_euclidiana(a1, a2, b1, b2). Observe que as linhas 12 e 13 devem ser alteradas para refletir os novos nomes dos parâmetros.
- Linhas 11 à 14: A função está definida no local incorreto. Ela precisa ser definida antes de sua chamada na linha 8 ou, melhor ainda, antes de qualquer outro comando diferente de importação do corpo principal do programa. Ou seja, o ideal seria definí-la na linha 2.
- Linha 14: O cálculo do quadrado da diferença dos valores y dos pontos A e B está incorreto (foi calculado a multiplicação por dois ao invés da potência de dois): distância = math.sqrt(delta_x**2 + delta_y**2)
- Linha 15: A função não está retornando um valor, mas o programa espera que seja retornada a distância euclidiana. Assim, é necessário acrescentar uma linha à função: return distância.

Programa completo e corrigido:

```
import math

def distância_euclidiana(a1, a2, b1, b2):
    delta_x = b1 - a1
    delta_y = b2 - a2
    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)
print("A distância entre A e B é de %.2f" % distancia_a_b)
```

Considerando os dados de entrada solicitados, o resultado do programa (desconsiderando os conteúdos impressos nas linhas que pedem os dados de entrada) será: "A distância entre A e B é de 4.47". Observe que qualquer valor entre 4 e 5, desde que formatado conforme especificado na chamada de função *print* (com duas casas decimais) será considerado correto.