# MAC0110 – Introdução à Computação
## Prova 1 – 19/04/2018 – BCC

**NOME** (EM LETRA DE FORMA LEGÍVEL):

**ASSINATURA**:

**No. USP**:

### Instruções

1. Não destaque as folhas deste caderno.
2. A prova pode ser feita a lápis.
3. A legibilidade também faz parte da nota!
4. A prova consta de 3 questões. Verifique antes de começar a prova se o seu caderno de questões está completo.
5. Não é permitido o uso de folhas avulsas para rascunho.
6. Não é necessário apagar rascunhos no caderno de questão mas especifique qual é a resposta e qual é o rascunho.
7. Só é permitido usar os recursos dados nas aulas até o dia desta prova e deve-se seguir todas as restrições dadas também.
8. A prova é sem consulta.

**Não escrever nesta parte da folha**

| Questão | Nota | Observação |
|---|---|---|
| 1 | | |
| 2 | | |
| 3 | | |
| Total | | |

## Boa Prova!

**Questão 1 (valor=1.5+1.5)**

a) Escreva uma função `probabilidade(N,dist,x)` em Python que recebe valores inteiros $N > 0$ e dist$> 0$ (com $N$ dígitos), representando uma roleta enviesada como no EP1, e um valor $x \in \{0, \ldots, N-1\}$, e calcula a probabilidade desse valor ser sorteado pela roleta. Para ajudar a memória, no EP1 a roleta enviesada é representada pelo número de códigos sorteados $N$ e por uma lista de pesos $p_0, p_1, \ldots, p_{N-1} \in \{0, \ldots, 9\}$ codificados em um inteiro dist$=p_{N-1} * 10^{N-1} + \cdots + p_1 * 10^1 + p_0 * 10^0$, onde os pesos definem a probabilidade de sorteio de cada valor $n \in \{0, \ldots, N-1\}$ através da expressão $\text{prob}(x = n) = \frac{p_n}{S}$ e $S = p_0 + p_1 + \cdots + p_{N-1}$. Por exemplo, se $N = 3$, dist$= 735$, as chamadas `probabilidade(N,dist,0)`, `probabilidade(N,dist,1)` e `probabilidade(N,dist,2)` devem devolver, respectivamente, os valores $\frac{p_0}{S} = \frac{5}{15} = 0.333\ldots$, $\frac{p_1}{S} = \frac{3}{15} = 0.2$ e $\frac{p_2}{S} = \frac{7}{15} = 0.466\ldots$.

b) Escreva um programa em Python que lê do teclado valores inteiros $N > 0$, dist$> 0$ e uma lista de inteiros entre 0 e $N-1$ terminada por -1, e calcula a probabilidade dessa sequência ser sorteada pela roleta enviesada representada por (N,dist). Lembre que a probabilidade de uma sequência de eventos independentes é o produto das probabilidades individuais, ou seja, a probabilidade da roleta sortear a sequência de números $x_1, x_2, \ldots, x_K$ será igual a $\prod_{i=1}^{k} \left[\text{prob}(x = x_i = n) = \frac{p_n}{S}\right]$. Por exemplo, se $N = 3$, dist$= 735$ e a sequência for $0, 2, 2, 1, 1, 1, 0$ então $S = 7 + 3 + 5 = 15$ e a probabilidade desse sorteio será

$$\frac{p_0}{S}\frac{p_2}{S}\frac{p_2}{S}\frac{p_1}{S}\frac{p_1}{S}\frac{p_1}{S}\frac{p_0}{S} = \frac{5}{15}\frac{7}{15}\frac{7}{15}\frac{3}{15}\frac{3}{15}\frac{3}{15}\frac{5}{15} = 0.0001935802469135803.$$

**Dicas:** (1) Você não precisa ler os pesos $p_n$ e construir o inteiro dist, pois ele será digitado todo de uma vez pelo usuário. (2) Diferentemente do EP1, você pode usar o operador ** para resolver essa questão. (3) Você pode resolver o item (b) sem ter feito o item (a), basta usar aquela função corretamente.

```
# parte (a)

def probabilidade(N,dist,x):
    # calcula a soma dos dígitos, guardando
    # o x-ésimo dígito menos significativo
    n = 0 # índice associado às potências de 10
    px = 0 # peso associado a x
    S = 0 # soma dos pesos
    while dist>0:
        if x==n: px = dist%10
        S += dist%10 # soma último dígito
        dist //= 10 # extrai último dígito
        n += 1 # avança índice
    return px/S

# parte (b)

N = int(input("Digite N: "))
dist = int(input("Digite dist: "))
x = int(input("Digite a sequência terminada por -1: "))
prob = 1 # produto das probabilidades
while x!=-1:
    prob *= probabilidade(N,dist,x)
    x = int(input(""))
print("A probabilidade de sorteio dessa sequência é",prob)
```

## Questão 2 (valor=3.0)

Escreva uma função em Python que ajude a gerenciar um sistema de elevadores de um prédio conforme a descrição a seguir. O prédio possui 3 elevadores representados por 3 valores inteiros e1, e2 e e3; se $e_n > 0$ isso significa que o elevador $n$ está parado no andar $e_n$, e se $e_n < 0$ isso significa que ele está se dirigindo ao andar abs($e_n$). Considere que os andares do prédio começam no 1, e que não se sabe a altura do prédio (o sistema será usado em muitos prédios diferentes). Sua função será acionada cada vez que um usuário chamar um elevador no andar X, devendo devolver o número do elevador (1, 2 ou 3) que irá atender aquele usuário, de acordo com o protótipo

```
def chamaelevador(X,e1,e2,e3):
    """ Calcula o elevador que irá atender o usuário no andar X.
    """
    # seu código aqui...
    return elevador # 1, 2 ou 3
```

A política de atendimento deve priorizar o elevador que estiver mais perto, dando preferência para os elevadores que estão parados; apenas se todos os elevadores estiverem em movimento, deve-se dar preferência para aquele vai parar mais perto.

```
def chamaelevador(X,e1,e2,e3):
    """ Calcula o elevador que irá atender o usuário no andar X.
    """

    # decide qual elevador parado está mais perto
    k = 0 # índice do elevador selecionado
    dist = -1 # menor distância ao andar X
              # dist<0 ==> min não inicializado
    if e1>0:
        k = 1
        dist = abs(X-e1)
    if e2>0 and (dist<0 or abs(X-e2)<dist):
        k = 2
        dist = abs(X-e2)
    if e3>0 and (dist<0 or abs(X-e3)<dist):
        k = 3
        dist = abs(X-e3)
    # se existe elevador parado devolve o mais próximo
    if k>0:
        return k

    # decide qual elevador em movimento está mais perto.
    # nesse caso, já sabemos que e1,e2,e3<0
    k = 1
    dist = abs(X-abs(e1))
    if abs(X-abs(e2))<dist:
        k = 2
        dist = abs(X-abs(e2))
    if abs(X-abs(e3))<dist:
        k = 3
        dist = abs(X-abs(e3))
    return k
```

## Questão 3 (valor=3.0+1.0)

a) Escreva uma função com protótipo

```
def triplaPitagoreana(N):
    """ Tenta encontrar a,b,c com 0<a<b<c e N == a+b+c,
        que satisfaçam a condição a**2+b**2 == c**2.
    """
    # seu código aqui...
    return encontrei, a, b, c
```

que recebe um inteiro $N > 0$ e percorre todas as triplas de valores inteiros $(a, b, c)$ com $0 < a < b < c$ e $N = a + b + c$, verificando se essas triplas satisfazem a condição $a^2 + b^2 = c^2$. Se isso ocorrer, sua função deve devolver `encontrei=True` e os valores encontrados de $a, b$ e $c$; do contrário, ela devolverá `encontrei=False` e $a = b = c = 0$. **Dica:** a solução correta usa apenas 2 laços, mas se você só souber resolver essa questão com 3 laços, resolva-a!

b) Escreva um programa que pede ao usuário um valor inteiro $K > 0$ e produz a lista de todos os valores de $N \in \{0, 1, \ldots, K\}$ que possuem triplas Pitagoreanas associadas. **Dica:** você pode resolver esse item mesmo sem ter feito o item (a), basta usar a função.

```
# parte (a)

def triplaPitagoreana(N):
    """ Tenta encontrar a,b,c com 0<a<b<c e N == a+b+c,
        que satisfaçam a condição a**2+b**2 == c**2.
    """
    # percorre a=1,2,...,N/3
    # ( se a>=N/3, então b>=N/3+1 e c>=N/3+2,
    #   logo a+b+c >= N+3 > N )
    a = 1
    while a<N/3:
        # percorre b=a+1,...,(N-a)/2
        # ( se b>=(N-a)/2, então c>=(N-a)/2+1,
        #   logo a+b+c >= a+(N-a)+1 > N )
        b = a+1
        while b<(N-a)/2:
            c = N-a-b
            if a**2+b**2==c**2:
                return True, a, b, c
            b += 1
        a += 1
    return False, 0, 0, 0

# parte (b)

K = int(input("Digite um inteiro K>0: "))
for N in range(K+1):
    encontrei, a, b, c = triplaPitagoreana(N)
    if encontrei:
        print(N,"=",a,"+",b,"+",c)
```