Universidade Federal de Goiás
Escola de Engenharia Elétrica, Mecânica e de Computação
Aluno: Edson Júnior Frota Silva Matrícula: 201515412
Engenharia de Computação

Laboratório de Inovação e Automação 1 Prof.: Cássio Vinhal

Exercícios referentes ao capítulo 10:

Exercicio_10.01:

```
t = [[1, 2], [3], [4, 5, 6]]
def nested_sum(t):
    soma = []
    for i in t:
        soma += i
    return sum(soma)
nested_sum(t)
```

Exercicio_10.02:

```
def cumsum(t):
   r = []
   for i in range(len(t)):
      soma = 0
      for j in range( i+1 ):
          soma += t[j]
      r.append(soma)
   return(r)
t = [1, 2, 3]
soma = cumsum(t)
print(soma)
```

Exercicio_10.03:

```
def middle(t):
   return t[1:-1]
t = [1, 2, 3, 4]
r = middle(t)
print(r)
```

Exercicio_10.04:

```
def chop(aux):
   aux.pop(0)
   aux.pop(-1)
aux = [1, 2, 3, 4]
cont = chop(aux)
print(cont)
```

```
print(aux)
```

Exercicio_10.05:

```
def is_sorted(aux):
    for kO in range(1, len(aux)):
        if not aux[kO-1] <= aux[kO]:
            return False
    return True
aux = [1, 2, 3]
print(is_sorted(aux))
cont1 = ['a', 'c']
print(is_sorted(cont1))
cont2 = ['b', 'a']
print(is_sorted(cont2))
```

Exercicio_10.06:

```
def is_anagram(x, y):
    aux = sorted(x)
    cont = sorted(y)
    if aux == cont :
        return True
    return False

aux = 'marrocos'
cont = 'socorram'
print(is_anagram(aux, cont))
```

Exercicio_10.07:

```
def has_duplicates(aux):
    for i in range(1, len(aux)):
        if aux[i-1] in aux[i:]:
            return True
    return False
x = [1, 2, 3, 4, 2]
y = ['a', 'c', 'a', 'j']
z = [3, 5, 1, 6, 7]
print(has_duplicates(x))
print(has_duplicates(y))
print(has_duplicates(z))
```

Exercicio_10.08:

```
import random
def gen_birthdays(n):
    t = []
    for i in range(n):
        t.append(random.randint(1,365))
```

```
    return t
def check_match(t):
    for i in range(len(t)):
        if t[i] in t[i+1:]:
            return True
    return False
def run_simulations(n_simulations, n_pessoas):
    count_matchs = 0
    for i in range(n_simulations):
        t = gen_birthdays(n_pessoas)
        if check_match(t): count_matchs += 1
    return count_matchs
ns = int(input('Insira o número de simulações que devem ser
rodadas: '))
np = int(input('Insira o número de pessoas por simulação: '))
print('')
matchs = run_simulations(ns, np)
p = (matchs/ns)*100
print(str(matchs) + ' matchs em ' + str(ns) + ' simulações com
pelo menos 1 match em cada')
print(str(p) + '% de simulações com pelo menos um match')
```

Exercicio_10.09:

```
import time
def construct_list_append():
    fin = open('/root/words.txt')
    r = []
    for line in fin:
        r.append(line.strip())
    fin.close()
    return r
def construct_list_noAppend():
    fin = open('/root/words.txt')
    r = []
    for line in fin:
        r = r + [line.strip()]
    fin.close()
    return r
start = time.time()
construct_list_append()
et = time.time() - start
print('Com append demorou ' + str(et) + ' segundos')
start = time.time()
construct_list_noAppend()
et = time.time() - start
print('Sem append demorou ' + str(et) + ' segundos')
```

Exercicio_10.10:

```
fin = open('/root/words.txt')
```

```
count = 0
def load_list():
    r = []
    for line in fin:
        r.append(line.strip())
    return r
def in_bisect(lista, word, min, max):
    i = (max+min)//2
    global count
    count += 1
    if max < min or not i in range(0, len(lista)):
        return None
    if word == lista[i]:
        return i
    elif word < lista[i]:
        return in_bisect(lista, word, min, i-1)
    elif word > lista[i]:
        return in_bisect(lista, word, i+1, max)
words = load_list()
word = str(input('Insira a palavra que deseja buscar: '))
index = in_bisect(words, word, 0, len(words))
if not index is None:
    print('A palavra se encontra no indice: ' + str(index))
else:
    print('A palavra não se encontra na lista')
print('Passos: ' + str(count))
```

Exercicio_10.11:

```
fin = open('/root/words.txt')
count = 0
def load_list():
    r = []
    for line in fin:
        r.append(line.strip())
    return r
def in_bisect(lista, word, min, max):
    i = (max+min)//2
    global count
    count += 1
    if max < min or not i in range(0, len(lista)):
        return None
    if word == lista[i]:
        return i
    elif word < lista[i]:
        return in_bisect(lista, word, min, i-1)
    elif word > lista[i]:
        return in_bisect(lista, word, i+1, max)
r = load_list()
for word in r:
    if not in_bisect(r, word[::-1], 0, len(r)) is None:
        print(word)
```

```
print(str(count) + ' comparações')
```

Exercicio_10.11:

```
fin = open('/root/words.txt')
count = 0
def load_list():
    r = []
    for line in fin:
        r.append(line.strip())
    return r
def in_bisect(lista, word, min, max):
    i = (max+min)//2
    global count
    count += 1
    if max < min or not i in range(0, len(lista)):
        return None
    if word == lista[i]:
        return i
    elif word < lista[i]:
        return in_bisect(lista, word, min, i-1)
    elif word > lista[i]:
        return in_bisect(lista, word, i+1, max)
def is_interlocked(word, passos=2):
    wp = word[::passos]
    wi = word[1::passos]
    if not in_bisect(r, wp, 0, len(r)) is None and not in_bisect(r,
wi, 0, len(r)) is None:
        print(wp + '   +   ' + wi)
        return True
    else:
        return False
r = load_list()
for e in r:
    if is_interlocked(e):
        print(e)
```

Edson Júnior Frota Silva
Engenharia de Computação - UFG