University of Texas at San Antonio
Department of Electrical and Computer Engineering
EE 5453: SW Engineering in Python, Matlab, and Android
Dr. Kelley, Milestone 1: Python Introduction
Professor Brian Kelley
Due Thursday, 02/06/2020: 11 pm

Place each problem solution in a separate folder and zip the collection of folders and upload to zipped files to Blackboard

Problem 1: 20 points

Rewrite the following Python script without the use of the break or continue command

```
y=1
for x in range(4,256,4):
   y = y*x
   if y > 512:
      break
   print y
```

```
y = 1
for x in range(4, 256, 4):
    y = y*x
    if y > 512:
        break
    print(y)  # Python 3.7 print
```

```
# defined states; execute till end of loop
y = 1
for x in range(4, 256, 4):
    y = y*x
    if y > 512:
        pass
    elif y <= 512:
        print(y)
```

```
# defined states; while execute only necessary loops with variable number of
iterations
y = 1
x = 4   # compound statement on one line separated by semicolon
breakloop = 0
while breakloop == 0:
    y = y * x
    x += 4
    if y > 512:
        breakloop = 1
    elif y <= 512:
        print(y)
```

```
4
32
384
4
32
384
4
32
384
```

Problem 2: 20 points

Rewrite the following Python script without the use of the break or continue command

```
#!/usr/bin/python

fruits=["apple","orange","banana","mango"]
for item in fruits:
   if item == "banana":
      continue  #continue with next iteration
   else:
      print item
```

```
#!/usr/bin/python

fruits = ["apple", "orange", "banana", "mango"]
for item in fruits:
    if item == "banana":
        continue  # continue with next iteration
    else:
        print(item)


# define the states explicitly
for item in fruits:
    if item == "banana":
        pass  # continue with next iteration
    elif item != "banana":
        print(item)
```

```
apple
orange
mango
apple
orange
mango
```

Problem 3: 30 points

Part 3A: Create a nested Python dictionary denoted *students*.

- The 1st Dictionary key is the ID-number (e.g. 11) that points to a 2nd dictionary as the value.

- The 2nd nested Dictionary should contain *name* and *grade* as key (e.g. 'name', 'grade') pointing to the *Student* and *Grade* as value (e.g. 'Bob', 2.5)

Student	ID	Grade
Bob	11	2.5
Mary	21	3.5
David	31	4.2
John	42	4.1
Alex	53	3.8

Part 3b: Create a python function named averageGrade that inputs the dictionary *students* and will compute and returns the average_grade

Create another python function named nameList that inputs the dictionary *students* and will print the name of students in alphabetical order

Create another python function named gradeList that inputs the dictionary *students* and will print the name of students in grade order, highest grade first to lowest grade last

Write a python script that invokes the averageGrade function and prints "The average grade is " average_grade

```
import numpy as np


def averageGrade(students):
    # This function computes the average grade
    sum = 0.0
    for key in students:
        sum = sum + students[key]['grade']
        average = sum / len(students)
    return average


students = {'11': {'name': 'Bob', 'grade': 2.5},
            '21': {'name': 'Mary', 'grade': 3.5},
            '31': {'name': 'David', 'grade': 4.2},
            '42': {'name': 'John', 'grade': 4.1},
            '53': {'name': 'Alex', 'grade': 3.8}}

import numpy as np


def averagegrade(students):
    # This function computes the average grade
    sum = 0.0
    for key in students:
```

```
        sum = sum + students[key]['grade']
        average = sum / len(students)
    return average


def ListSort1(students):
    # This function computes the average grade
    x = list()
    for key in students:
        x.append(students[key]['name'])
    y = sorted(x)
    return sorted(y)

def ListSort2(students):
    # This function computes the average grade
    x = list()
    y = list()
    z = list()
    V1 = list()
    for key in students:
        x.append(students[key]['name'])
        y.append(-students[key]['grade'])
    z = np.argsort(y)
    for d in z:
        V1.append(x[d])
    return V1

#b = np.zeros((2,3), dtype=complex)
avg = averagegrade(students)
print(ListSort1(students))
print(ListSort2(students))
print("The average grade is:", avg)

['Alex', 'Bob', 'David', 'John', 'Mary']
['David', 'John', 'Alex', 'Mary', 'Bob']
The average grade is: 3.6199999999999997
```

Problem 4: 30 points

Part A: Create Python function called displayFruits that will

- take in a Python list of fruits
- add a 'mango' to the end of the list
- prints, "There are" SizeOfList "fruits in the list containing [fruit1, fruit2,….fruitN, mango]"
    - SizeOfList *must be* the number of *unique names* in the list irrespective of the case (e.g. orange and Orange and ORAnge are not uniqe)

```
#!/usr/bin/python
fruitylonglist = ["apple", "orange", "Orange", "ORAnge", "banana", "MAngo"]


# define the states explicitly
def displayfruits(fruits):
```

```
    fruits.append("mango")
    x = list()
    for item1 in fruits:
        x.append(item1.lower())

    fruitlist = list()
    for item2 in x:
        if len(fruitlist) == 0:
            fruitlist.append(item2)
        elif len(fruitlist) > 0:
            uniqueelement = 1
            for item3 in fruitlist:
                if item3 == item2:
                    uniqueelement = 0
            if uniqueelement == 1:
                fruitlist.append(item2)
    return fruitlist


fruitout = displayfruits(fruitylonglist)
SizeOfList = len(fruitout)
print("There are", SizeOfList, "fruits in the list containing", fruitout)
```

```
There are 4 fruits in the list containing ['apple', 'orange', 'banana',
'mango']
```