Take screenshots of the completed programming exam (python code and output in python shell), paste them into a word document and upload it. Please submit your assignment with the naming convention of Lastname_Firstname_Practice_Programming_Exam2.docx.

<u>All Output</u>

```
Python Shell: Wing
Python Shell
Commands execute without debug. Use arrow keys for history.
Options
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "help", "copyright", "credits" or "license" for more information.
>>> [evaluate Nguyen_Albert_Practice_Programming_Exam_2.py]
********** Q1 **************************************************
Please Enter the side length for the square: 7
******* *******
******* *     *
******* *     *
******* *     *
******* *     *
******* *     *
******* *******
********** Q2 **************************************************
For list x containing [1, 4, 9, 16, 9, 7, 4, 9, 11] and list y containing [11, 1, 4, 9, 16, 9, 7, 4, 9]
The lists are identical
For list x containing [1, 4, 9, 16, 9, 7, 4, 9, 11] and list y containing [11, 11, 7, 9, 16, 4, 1, 4, 9]
The lists are not identical
********** Q3 **************************************************
allSame of 1, 2, and 3 is False
allSame of 3, 2, and 1 is False
allSame of 1, 3, and 2 is False
allSame of 2, 2, and 2 is True
allDifferent of 1, 2, and 3 is True
allDifferent of 3, 2, and 1 is True
allDifferent of 1, 3, and 2 is True
allDifferent of 2, 2, and 2 is False
sorted of 1, 2, and 3 is True
sorted of 3, 2, and 1 is False
sorted of 1, 3, and 2 is False
sorted of 2, 2, and 2 is True
>>>
```

10 points

Organization of the code such as proper commenting out, meaningful variable names, declaring and calling functions such as calling helper functions in main function, following PEP 8 design, clean code, submitting the word document with properly pasted python code and output etc.

30 points

Q1. Write a program that reads an integer and displays, using asterisks, a filled and hollow square, placed next to each other.

Here is a sample dialog

Please Enter the side length for the square: 7

```
******* *******
```

```
******* *      *

******* *      *

******* *      *

******* *      *

******* *      *

******* *******
```

Q1 Code

```python
# Q1
def sideLengthSquare() :
    length = input("Please Enter the side length for the square: ")
    lengthInt = int(length)
    print("*" * lengthInt, "*" * lengthInt)
    for i in range(1, (lengthInt - 1)) :
        print("*" * lengthInt, end = " ")
        print("*", end="")
        print(" " * (lengthInt - 2), end="")
        print("*")
    print("*" * lengthInt, "*" * lengthInt)


# Q2
def sameItems(x,y) :
    sortedX = sorted(x)
    sortedY = sorted(y)
    listChecker = []
    for i in range(len(x)) :
```

Q1 Cropped Output

Python Shell: Wing

Python Shell

Commands execute without debug. Use arrow keys for history.

```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "help", "copyright", "credits" or "license" for more information.
>>> [evaluate Nguyen_Albert_Practice_Programming_Exam_2.py]
********** Q1 ************************************************
Please Enter the side length for the square: 7
******* *******
******* *     *
******* *     *
******* *     *
******* *     *
******* *     *
******* *******
```

Q1 Code_Actual/Call

```
def sideLengthSquare() :
    length = input("Please Enter the side length for the square: ")
    lengthInt = int(length)
    print("*" * lengthInt, "*" * lengthInt)
    for i in range(1, (lengthInt - 1)) :
        print("*" * lengthInt, end = " ")
        print("*", end="")
        print(" " * (lengthInt - 2), end="")
        print("*")
    print("*" * lengthInt, "*" * lengthInt)


def main() :  # cropped to exclude Q2 and Q3 (see Q2 and Q3 for respective calls)
    # Q1 Calls
    print("*" * 10, "Q1", "*" * 50)
    sideLengthSquare()
```

30 points

Q2. Write a function def sameItems(x,y) that checks whether two lists have the same elements in some order, with the same multiplicities.

1 4 9 16 9 7 4 9 11

and

11 1 4 9 16 9 7 4 9

would be considered identical, but

1 4 9 16 9 7 4 9 11

and

would not.

Q2 Code

```python


# Q2
def sameItems(x,y) :
    sortedX = sorted(x)
    sortedY = sorted(y)
    listChecker = []
    for i in range(len(x)) :
        if sortedX[i] == sortedY[i] :
            listChecker.append("True")
        else:
            listChecker.append("False")
    print("For list x containing %s and list y containing %s" % (x, y))
    if len(x) != len(y):
        print("The lists are not identical")
    elif "False" in listChecker :
        print("The lists are not identical")
    else :
        print("The lists are identical")

```

Here is a sample dialog

List 1 is [1, 2, 1, 1, 2]

List 2 is [1, 1, 1, 2, 2]

The lists contain the same elements: True

<u>Q2 Cropped Output</u>

```
********** Q2 ***************************************************
For list x containing [1, 4, 9, 16, 9, 7, 4, 9, 11] and list y containing [11, 1, 4, 9, 16, 9, 7, 4, 9]
The lists are identical
For list x containing [1, 4, 9, 16, 9, 7, 4, 9, 11] and list y containing [11, 11, 7, 9, 16, 4, 1, 4, 9]
The lists are not identical
```

<u>Q2 Code Actual/Call</u>

```python
def sameItems(x,y) :
    sortedX = sorted(x)
    sortedY = sorted(y)
    listChecker = []
    for i in range(len(x)) :
        if sortedX[i] == sortedY[i] :
            listChecker.append("True")
        else:
            listChecker.append("False")
    print("For list x containing %s and list y containing %s" % (x, y))
    if len(x) != len(y):
```

```
        print("The lists are not identical")
    elif "False" in listChecker :
        print("The lists are not identical")
    else :
        print("The lists are identical")

def main() :  # cropped to exclude Q1 and Q3 (see Q1 and Q3 for respective calls)
    # Q2 Calls
    print("*" * 10, "Q2", "*" * 50)
    # First Call
    x = [1, 4, 9, 16, 9, 7, 4, 9, 11]
    y = [11, 1, 4, 9, 16, 9, 7, 4, 9]
    sameItems(x,y)
    y = [11, 11, 7, 9, 16, 4, 1, 4, 9]
    # Second Call
    sameItems(x,y)
```

30 points

Q3. Write the following functions and provide a python program to test them.

a. def allSame(a, b, c) (returning true if the arguments are all the same)

b. def allDifferent(a, b, c) (returning true if the arguments are all different)

c. def sorted(a, b, c) (returning true if the arguments are sorted, with the smallest one coming first)

Here is a sample dialog

allSame of 1, 2, and 3 is False

allSame of 3, 2, and 1 is False

allSame of 1, 3, and 2 is False

allSame of 2, 2, and 2 is True

allDifferent of 1, 2, and 3 is True

allDifferent of 3, 2, and 1 is True

allDifferent of 1, 3, and 2 is True

allDifferent of 2, 2, and 2 is False

sorted of 1, 2, and 3 is True

sorted of 3, 2, and 1 is False

sorted of 1, 3, and 2 is False

sorted of 2, 2, and 2 is True

Q3 Code

Nguyen_Albert_Practice_Programming_Exam_2.py (C:\Users\anguyen798\Documents\CS131-47853-homework\Chapter 6): Wing

File Edit Source Debug Tools Window Help

Lab7B.py | Nguyen_Albert_Practice_Programming_Exam_2.py *

allSame

```python

# Q3
# a
def allSame(a, b, c) :
    if a == b and b == c and a == c :
        print("allSame of %s, %s, and %s is True" % (a, b, c))
        return True
    else :
        print("allSame of %s, %s, and %s is False" % (a, b, c))
        return False


def allSameMain() :
    allSame(1,2,3)
    allSame(3,2,1)
    allSame(1,3,2)
    allSame(2,2,2)


# b
def allDifferent(a, b, c) :
    if a != b and b != c and a != c :
        print("allDifferent of %s, %s, and %s is True" % (a, b, c))
        return True
    else :
        print("allDifferent of %s, %s, and %s is False" % (a, b, c))
        return False

def allDifferentMain() :
    allDifferent(1,2,3)
    allDifferent(3,2,1)
    allDifferent(1,3,2)
    allDifferent(2,2,2)


# c
def sorted_(a, b, c) :
    if a < b and b < c:
        print("sorted of %s, %s, and %s is True" % (a, b, c))
        return True
    elif a == b == c:
        print("sorted of %s, %s, and %s is True" % (a, b, c))
        return True
    else :
        print("sorted of %s, %s, and %s is False" % (a, b, c))
        return False

def sortedMain() :
    sorted_(1,2,3)
    sorted_(3,2,1)
    sorted_(1,3,2)
    sorted_(2,2,2)


```

Q3 Cropped Output

```
********** Q3 ************************************************
allSame of 1, 2, and 3 is False
allSame of 3, 2, and 1 is False
allSame of 1, 3, and 2 is False
allSame of 2, 2, and 2 is True
allDifferent of 1, 2, and 3 is True
allDifferent of 3, 2, and 1 is True
allDifferent of 1, 3, and 2 is True
allDifferent of 2, 2, and 2 is False
sorted of 1, 2, and 3 is True
sorted of 3, 2, and 1 is False
sorted of 1, 3, and 2 is False
sorted of 2, 2, and 2 is True
>>>
```

Q2 Code Actual/Call

```
# Q3
# a
def allSame(a, b, c) :
    if a == b and b == c and a == c :
        print("allSame of %s, %s, and %s is True" % (a, b, c))
        return True
    else :
        print("allSame of %s, %s, and %s is False" % (a, b, c))
        return False


def allSameMain() :
    allSame(1,2,3)
    allSame(3,2,1)
    allSame(1,3,2)
    allSame(2,2,2)


# b
def allDifferent(a, b, c) :
    if a != b and b != c and a != c :
        print("allDifferent of %s, %s, and %s is True" % (a, b, c))
```

```
        return True
    else :
        print("allDifferent of %s, %s, and %s is False" % (a, b, c))
        return False

def allDifferentMain() :
    allDifferent(1,2,3)
    allDifferent(3,2,1)
    allDifferent(1,3,2)
    allDifferent(2,2,2)


# c
def sorted_(a, b, c) :
    if a < b and b < c:
        print("sorted of %s, %s, and %s is True" % (a, b, c))
        return True
    elif a == b == c:
        print("sorted of %s, %s, and %s is True" % (a, b, c))
        return True
    else :
        print("sorted of %s, %s, and %s is False" % (a, b, c))
        return False

def sortedMain() :
    sorted_(1,2,3)
    sorted_(3,2,1)
    sorted_(1,3,2)
    sorted_(2,2,2)
```

def main() : # cropped to exclude Q1 and Q2 (see Q1 and Q2 for respective calls)

```
    #Q3 Calls
    print("*" * 10, "Q3", "*" * 50)
    allSameMain()
    allDifferentMain()
    sortedMain()
```