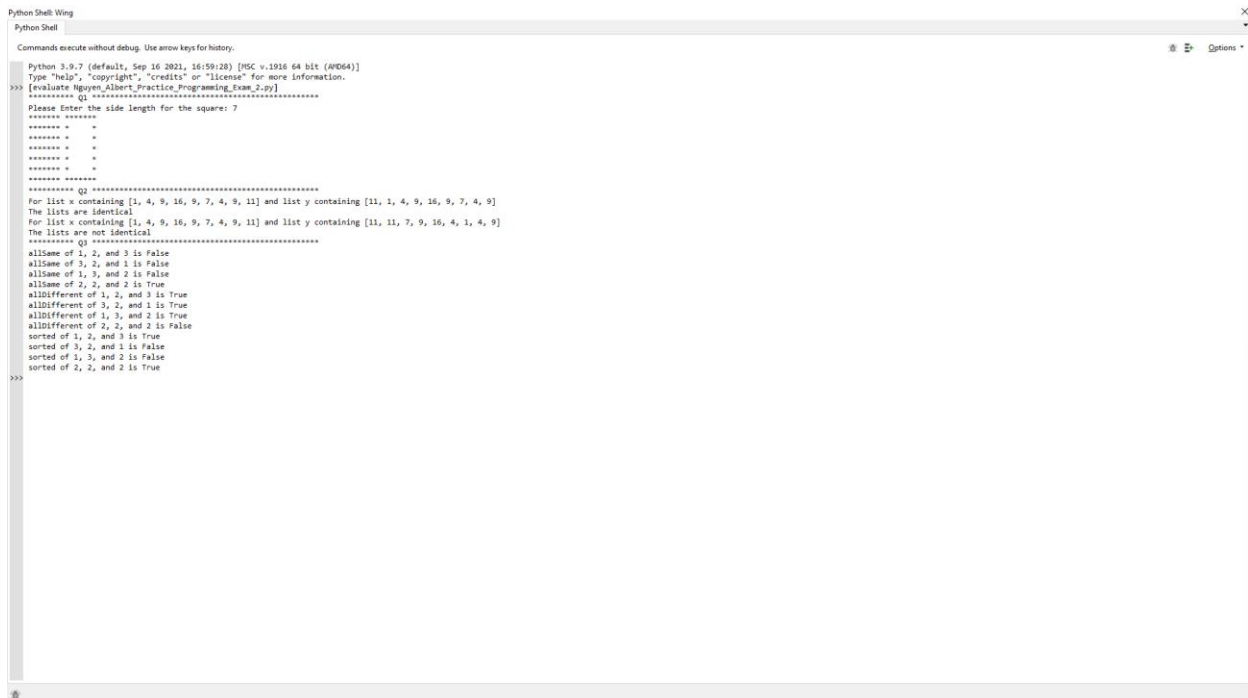


Take screenshots of the completed programming exam (**python code and output in python shell**), paste them into a word document and upload it. Please submit your assignment with the naming convention of Lastname_Firstname_Practice_Programming_Exam2.docx.

All 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
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
***** 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

```

1  I
2  # Q1
3  def sideLengthSquare() :
4      length = input("Please Enter the side length for the square: ")
5      lengthInt = int(length)
6      print("*" * lengthInt, "*" * lengthInt)
7      for i in range(1, (lengthInt - 1)) :
8          print("*" * lengthInt, end = " ")
9          print("*", end="")
10         print(" " * (lengthInt - 2), end="")
11         print("*")
12     print("*" * lengthInt, "*" * lengthInt)
13
14
15 # Q2
16 def sameItems(x,y) :
17     sortedX = sorted(x)
18     sortedY = sorted(y)
19     listChecker = []
20     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

```
Nguyen_Albert_Practice_Programming_Exam_2.py (C:\Users\anguyen790\Documents\CS131-47833-homework\Chapter 6\ Wing
File Edit Source Debug Tools Window Help
Lab7B.py Nguyen_Albert_Practice_Programming_Exam_2.py
13
14
15 # Q2
16 def sameItems(x,y) :
17     sortedX = sorted(x)
18     sortedY = sorted(y)
19     listChecker = []
20     for i in range(len(x)) :
21         if sortedX[i] == sortedY[i] :
22             listChecker.append("True")
23         else:
24             listChecker.append("False")
25     print("For list x containing %s and list y containing %s" % (x, y))
26     if len(x) != len(y):
27         print("The lists are not identical")
28     elif "False" in listChecker :
29         print("The lists are not identical")
30     else :
31         print("The lists are identical")
32
```

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

Q2 Cropped Output

```
***** 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
```

Q2 Code Actual/Call

```
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.

- def allSame(a, b, c) (returning true if the arguments are all the same)
- def allDifferent(a, b, c) (returning true if the arguments are all different)
- 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 2, 2, and 2 is True

```
Nguyen_Albert_Practice_Programming_Exam_2.py (C:\Users\anguyen798\homework\Chapter 6): Wing
File Edit Source Debug Tools Window Help
[Icons: File Explorer, Run, Stop, Breakpoint, etc.]
Lab7B.py Nguyen_Albert_Practice_Programming_Exam_2.py*
allSame v
33
34 # Q3
35 # a
36 def allSame(a, b, c) :
37     if a == b and b == c and a == c :
38         print("allSame of %s, %s, and %s is True" % (a, b, c))
39         return True
40     else :
41         print("allSame of %s, %s, and %s is False" % (a, b, c))
42         return False
43
44
45 def allSameMain() :
46     allSame(1,2,3)
47     allSame(3,2,1)
48     allSame(1,3,2)
49     allSame(2,2,2)
50
51
52 # b
53 def allDifferent(a, b, c) :
54     if a != b and b != c and a != c :
55         print("allDifferent of %s, %s, and %s is True" % (a, b, c))
56         return True
57     else :
58         print("allDifferent of %s, %s, and %s is False" % (a, b, c))
59         return False
60
61 def allDifferentMain() :
62     allDifferent(1,2,3)
63     allDifferent(3,2,1)
64     allDifferent(1,3,2)
65     allDifferent(2,2,2)
66
67
68 # c
69 def sorted_(a, b, c) :
70     if a < b and b < c:
71         print("sorted of %s, %s, and %s is True" % (a, b, c))
72         return True
73     elif a == b == c:
74         print("sorted of %s, %s, and %s is True" % (a, b, c))
75         return True
76     else :
77         print("sorted of %s, %s, and %s is False" % (a, b, c))
78         return False
79
80 def sortedMain() :
81     sorted_(1,2,3)
82     sorted_(3,2,1)
83     sorted_(1,3,2)
84     sorted_(2,2,2)
85
86
```

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()

```