CSC 110 Midterm 2 Study Guide

Question 1

Loop table

Give the function below:

```
def double_nested(lists):
  for i in range(len(lists)):
    for j in range(len(lists[i])):
      lists[i][j] *= 2
  return lists
```

Complete the `loop table` below with the corresponding values of `i`, `j`, `len(lists[i])` and `lists` for the following function call:

```
double_nested([[0, 1, 3, 1], [2]])
```

RESPONSE FOR QUESTION 1:

i	j	len(lists[i])	lists
			[[_____, _____, _____, _____], [_____]]
			[[_____, _____, _____, _____], [_____]]
			[[_____, _____, _____, _____], [_____]]
			[[_____, _____, _____, _____], [_____]]
			[[_____, _____, _____, _____], [_____]]

Question 2

Loop table

Give the function below:

```
def reverse_strings_nested(strings):
  for i in range(len(strings)):
    for j in range(len(strings[i])):
      strings[i][j] = strings[i][j][::-1]
  return strings
```

Complete the `loop table` below with the corresponding values of `i`, `j`, `len(lists[i])` and `lists` for the following function call:

```
reverse_strings_nested([["war", "peek"], ["dog"], ["pets", "snug", "net"]])
```

RESPONSE FOR QUESTION 2:

i	j	len(strings[i])	strings[i][j]	strings[i][j][::-1]	strings
					[[_____, _____], [_____], [_____, _____, _____]]
					[[_____, _____], [_____], [_____, _____, _____]]
					[[_____, _____], [_____], [_____, _____, _____]]
					[[_____, _____], [_____], [_____, _____, _____]]
					[[_____, _____], [_____], [_____, _____, _____]]
					[[_____, _____], [_____], [_____, _____, _____]]

Question 3

Loop table

Give the function below:

```
def nested_min(lists):
  min = None
  for i in range(len(lists)):
    for j in range(len(lists[i])):
      if min == None or lists[i][j] < min:
        min = lists[i][j]
  return min
```

Complete the `loop table` below with the corresponding values of `i`, `j`, `len(lists[i])`, `lists[i][j]` and `min` for the following function call:

```
nested_min([[4, 3, 1, 0], [], [7, 2, 1]])
```

RESPONSE FOR QUESTION 3:

i	j	len(lists[i])	lists[i][j]	min

Lists

Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be `ERROR`)

```
numbers = [1, 2]
numbers.append(5)
numbers.append(0)
numbers
```

```
numbers = [1, 2]
numbers.remove(1)
numbers
```

```
numbers = [1, 2]
numbers.pop(1)
numbers
```

```
numbers = [1, 2]
numbers[1]
numbers
```

```
numbers = [2, 3, 20, 1, 2, 40, 2]
numbers[7]
```

```
numbers = [2, 3, 20]
numbers[3] = 10
numbers
```

Dictionaries

Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be `ERROR`)

```
counts = {"a": 1, "b": 2}
counts[1] = "c"
counts
```

```
counts = {"a": 1, "b": 2}
counts["a"] = "c"
counts
```

```
counts = {"a": 1}
counts.append("b": 2)
counts
```

Tuples

Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be `ERROR`)

```
names = ("Ann", "Philipp", "Beatrice", "Paul")
names[3]
```

```
names = ("Ann", "Philipp", "Beatrice", "Paul")
names[4]
```

```
names = ("Ann", "Philipp", "Beatrice", "Paul")
names[1]
```

```
names = ("Ann", "Philipp", "Beatrice", "Paul")
names[1] = "Peter"
```

Files

Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be `ERROR`)

names.txt

```
Marcellin Burke
Albrecht Libby
Niki Zoey
Braxton Marvin
Marvin Brown
Ann Brown
```

```
f = open("names.txt", "r")
f.write("Ann\n")
f.close()
```

```
f = open("names.txt", "r")
names = []
for line in f:
  names.append(line)
f.close()
names
```

```
f = open("names.txt", "r")
names = []
for line in f:
  name = line.strip()
  names.append(name)
f.close()
names
```

```
f = open("names.txt", "r")
names = []
for line in f:
  line_names = line.strip().split(" ")
  for n in line_names:
    names.append(n)
f.close()
names
```

```
f = open("names.txt", "r")
names = []
for line in f:
  line_names = line.split(" ")
  for n in line_names:
    names.append(n)
f.close()
names
```

```
f = open("names.txt", "r")
names = {}
for line in f:
  line_names = line.strip().split(" ")
  for n in line_names:
    if n in names:
      names[n] += 1
    else:
      names[n] = 1
f.close()
names
```

File reading and writing

See the python code and the contents of the file name `data.txt`. The python code writes content to a file named `result.txt`. You must determine what the contents of `result.txt` will be after the code runs. Put your answer in the response box.

data.txt

```
one,two,three,four
five,six
seven,eight,nine
ten,eleven,twelve,thirteen,fourteen
```

```
def is_acceptable(x):
    if len(x) > 5:
      return True
    return False

def write_result(input_filename, output_filename):
    data = open(input_filename, 'r')
    result = open(output_filename, 'w')
    for line in data:
       numbers = line.strip().split(',')
       for n in numbers:
           if is_acceptable(n):
              result.write(n + ',')
    data.close()
    result.close()

def main():
   write_result('data.txt', 'result.txt')

main()
```

KEY

Question 1

Loop table

Give the function below:

```
def double_nested(lists):
  for i in range(len(lists)):
    for j in range(len(lists[i])):
      lists[i][j] *= 2
  return lists
```

Complete the `loop table` below with the corresponding values of `i`, `j`, `len(lists[i])` and `lists` for the following function call:

```
double_nested([[0, 1, 3, 1], [2]])
```

RESPONSE FOR QUESTION 1:

i	j	len(lists[i])	lists
0	0	4	[[0, 1, 3, 1], [2]]
0	1	4	[[0, 2, 3, 1], [2]]
0	2	4	[[0, 2, 6, 1], [2]]
0	3	4	[[0, 2, 6, 2], [2]]
1	0	1	[[0, 2, 6, 2], [4]]

Question 2

Loop table

Give the function below:

```
def reverse_strings_nested(strings):
  for i in range(len(strings)):
    for j in range(len(strings[i])):
      strings[i][j] = strings[i][j][::-1]
  return strings
```

Complete the `loop table` below with the corresponding values of `i`, `j`, `len(lists[i])` and `lists` for the following function call:

```
reverse_strings_nested([["war", "peek"], ["dog"], ["pets", "snug", "net"]])
```

RESPONSE FOR QUESTION 2:

i	j	len(strings[i])	strings[i][j]	strings[i][j][::-1]	strings
0	0	2	war	raw	[["raw", "peek"], ["dog"], ["pets", "snug", "net"]]
0	1	2	peek	keep	[["raw", "keep"], ["dog"], ["pets", "snug", "net"]]
1	0	1	dog	god	[["raw", "keep"], ["god"], ["pets", "snug", "net"]]
2	0	3	pets	step	[["raw", "keep"], ["god"], ["step", "snug", "net"]]
2	1	3	snug	guns	[["raw", "keep"], ["god"], ["step", "guns", "net"]]
2	2	3	net	ten	[["raw", "keep"], ["god"], ["step", "guns", "ten"]]

Question 3

Loop table

Give the function below:

```
def nested_min(lists):
  min = None
  for i in range(len(lists)):
    for j in range(len(lists[i])):
      if min == None or lists[i][j] < min:
        min = lists[i][j]
  return min
```

Complete the `loop table` below with the corresponding values of `i`, `j`, `len(lists[i])`, `lists[i][j]` and `min` for the following function call:

```
nested_min([[4, 3, 1, 0], [], [7, 2, 1]])
```

RESPONSE FOR QUESTION 3:

i	j	len(lists[i])	lists[i][j]	min
0	0	4	4	4
0	1	4	3	3
0	2	4	1	1
0	3	4	0	0
1	-	0	-	0
2	0	3	7	0
2	1	3	2	0
2	2	3	1	0

Lists

Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be `ERROR`)

```
numbers = [1, 2]
numbers.append(5)
numbers.append(0)
numbers
```

```
## [1, 2, 5, 0]
```

```
numbers = [1, 2]
numbers.remove(1)
numbers
```

```
## [2]
```

```
numbers = [1, 2]
numbers.pop(1)
```

```
## 2
```

```
numbers
```

```
## [1]
```

```
numbers = [1, 2]
numbers[1]
```

```
## 2
```

```
numbers
```

```
## [1, 2]
```

```
numbers = [2, 3, 20, 1, 2, 40, 2]
numbers[7]
```

ERROR

```
numbers = [2, 3, 20]
numbers[3] = 10
numbers
```

ERROR

Dictionaries

Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be `ERROR`)

```
counts = {"a": 1, "b": 2}
counts[1] = "c"
counts
```

```
## {'a': 1, 'b': 2, 1: 'c'}
```

```
counts = {"a": 1, "b": 2}
counts["a"] = "c"
counts
```

```
## {'a': 'c', 'b': 2}
```

```
counts = {"a": 1}
counts.append("b": 2)
counts
```

ERROR

Tuples

Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be `ERROR`)

```
names = ("Ann", "Philipp", "Beatrice", "Paul")
names[3]
```

```
## 'Paul'
```

```
names = ("Ann", "Philipp", "Beatrice", "Paul")
names[4]
```

ERROR

```
names = ("Ann", "Philipp", "Beatrice", "Paul")
names[1]
```

```
## 'Philipp'
```

```
names = ("Ann", "Philipp", "Beatrice", "Paul")
names[1] = "Peter"
```

ERROR

Files

Evaluate the code

Evaluate the code below (when the code throws an error, the answer should be `ERROR`)

names.txt

```
Marcellin Burke
Albrecht Libby
Niki Zoey
Braxton Marvin
Marvin Brown
Ann Brown
```

```
f = open("names.txt", "r")
f.write("Ann\n")
f.close()
```

ERROR

```
f = open("names.txt", "r")
names = []
for line in f:
  names.append(line)
f.close()
names
```

```
## ['Marcellin Burke\n', 'Albrecht Libby\n', 'Niki Zoey\n', 'Braxton Marvin\n', 'Marvin Brown\n', 'Ann
```

```
f = open("names.txt", "r")
names = []
for line in f:
  name = line.strip()
  names.append(name)
f.close()
names
```

```
## ['Marcellin Burke', 'Albrecht Libby', 'Niki Zoey', 'Braxton Marvin', 'Marvin Brown', 'Ann Brown']
```

```
f = open("names.txt", "r")
names = []
for line in f:
  line_names = line.strip().split(" ")
  for n in line_names:
    names.append(n)
f.close()
names
```

```
## ['Marcellin', 'Burke', 'Albrecht', 'Libby', 'Niki', 'Zoey', 'Braxton', 'Marvin', 'Marvin', 'Brown',
```

```
f = open("names.txt", "r")
names = []
for line in f:
  line_names = line.split(" ")
  for n in line_names:
    names.append(n)
f.close()
names
```

```
## ['Marcellin', 'Burke\n', 'Albrecht', 'Libby\n', 'Niki', 'Zoey\n', 'Braxton', 'Marvin\n', 'Marvin', '
```

```
f = open("names.txt", "r")
names = {}
for line in f:
  line_names = line.strip().split(" ")
  for n in line_names:
    if n in names:
      names[n] += 1
    else:
      names[n] = 1
f.close()
names
```

```
## {'Marcellin': 1, 'Burke': 1, 'Albrecht': 1, 'Libby': 1, 'Niki': 1, 'Zoey': 1, 'Braxton': 1, 'Marvin'
```

File reading and writing

See the python code and the contents of the file name `data.txt`. The python code writes content to a file named `result.txt`. You must determine what the contents of `result.txt` will be after the code runs. Put your answer in the response box.

data.txt

```
one,two,three,four
five,six
seven,eight,nine
ten,eleven,twelve,thirteen,fourteen
```

```
def is_acceptable(x):
    if len(x) > 5:
      return True
    return False

def write_result(input_filename, output_filename):
    data = open(input_filename, 'r')
    result = open(output_filename, 'w')
    for line in data:
       numbers = line.strip().split(',')
       for n in numbers:
           if is_acceptable(n):
              result.write(n + ',')
    data.close()
    result.close()

def main():
   write_result('data.txt', 'result.txt')

main()
```

result.txt

```
eleven,twelve,thirteen,fourteen,
```