The final will consist of 10 multiple choice problems worth 4 points each and 3 programming problems worth 20 points each.

For each multiple choice problem, select the letter that gives the output of the given code.

Note: the argument of the print statement "end = ' '" make successive executions of the print statement print on one line, separated by a space.

Problem 1

def threeTimesIsTrue(s):
    trueCount = 0
    for i in range(0, len(s)-1):
        if s[i] == s[i+1]:
            trueCount += 1
            print('True', end = ' ')
    if trueCount >= 3:
        return True
    else:
        return False

print(threeTimesIsTrue('aaabb'))

a. True True True
b. True True False
c. True True True True
d. True True True False
e. None of the above

Problem 2

def shortSorter(wordList):
    shortSort = []
    for i in wordList[0:len(wordList)-1]:
        if len(wordList[i]) <= len(wordList[i+1]):
            shortSort.append(i)
    return shortSort
dec21 = ['the', 'day', 'is', 'short']
print(shortSorter(dec21))

a. ['the', 'day']
b. ['the', 'day', 'short']
c. ['the', 'day', 'is', 'short']
d. TypeError: list indices must be integers, not str
e. None of the above
Problem 3
s = 'ahemahemahem'
print(s[4:9] + s[9:])
a. mahemahe
b. maheahem
c. ahemaahem
d. ahemahem
e. None of the above

Problem 4
utensils = ['fork', 'spoon', 'ladel', 'can opener']
hasO = []
for utensil in utensils:
    if 'o' in utensil:
        hasO.append(utensil)
print(hasO)
a. ['fork', 'spoon']
b. ['fork', 'spoon', 'can opener']
c. ['fork', 'spoon', 'spoon', 'can opener']
d. ['fork', 'spoon', 'opener']
e. None of the above

Problem 5
def vowelCheck(testStr):
    vowels = ['a', 'e', 'i', 'o', 'u']
    for letter in testStr:
        if letter not in vowels:
            return "no " + letter
        if len(testStr) == 0:
            return "no letters"
    else:
        return "no vowels"
s = 'even'
print(vowelCheck(s))
a. no a
a. no v
c. no v no n
d. SyntaxError: hanging else
e. None of the above
Problem 6

currency = [{'US': '$', 'Japan': 'yen', 'UK': 'pound'}, 'Euro', 'scrip']
print(currency[:2])

a. {'US': '$', 'Japan': 'yen'}
b. [{'US': '$', 'Japan': 'yen'}]
c. [{'US': '$', 'Japan': 'yen'}, 'Euro']
d. [{'US': '$', 'UK': 'pound', 'Japan': 'yen'}, 'Euro']
e. None of the above

Problem 7

menu = {1: 'fruit', 2: ['pasta', 'sauce'], 3: 'cake'}
print(menu[2])

a. SyntaxError: invalid syntax
b. 'cake'
c. {3: 'cake'}
d. ['pasta', 'sauce']
e. None of the above

Problem 8

def freq(s):
    splitLine = s.split()
    count = {}
    for word in splitLine:
        if word in count:
            count[word] += 1
        else:
            count[word] = 1
    leastFreq = splitLine[0]
    for word in count:
        if count[word] < count[leastFreq]:
            leastFreq = word
    return leastFreq

seussLine = 'I meant what I said and I said what I meant'
print(freq(seussLine))

a. I
b. and
c. KeyError: 'I'
d. ""
e. None of the above
Problem 9

\[
\begin{align*}
T &= True \\
F &= False \\
bools &= [T \text{ and not } T, F \text{ and not } F, T \text{ or not } T, F \text{ or not } F] \\
trueCount &= 0 \\
\text{for expr in bools:} \\
&\quad \text{if expr == True:} \\
&\quad \quad \text{trueCount} += 1 \\
\text{print(trueCount)}
\end{align*}
\]

a. 1
b. 2
c. 3
d. 4
e. None of the above

Problem 10

\[
\begin{align*}
def \text{testReturn}(s, t, i): \\
&\quad \text{if s.count(t) } \leq i: \\
&\quad \quad \text{return}(\"\leq\") \\
&\quad \text{if s.count(t) } \geq i: \\
&\quad \quad \text{return}(\"\geq\") \\
&\quad \text{if s.count(t) } == i: \\
&\quad \quad \text{return}(\"==\") \\
&\quad \text{return}(t)
\end{align*}
\]

\[
str0 = \text{'Of the bells, bells, bells, bells,'} \\
str1 = \text{'bells'}
\]

print(testReturn(str0, str1, 4))

a. <=
b. '<=', '>=', '==', 'bells'
c. ==
d. bells
e. None of the above

Problem 11

\[
\begin{align*}
\text{places} &= \{\text{'Union': 'Union'}, \text{'Perth': 'Milburn'}, \text{'Essex': ['Bloomfield', 'West Orange']}\} \\
\text{print(places['Union'])}
\end{align*}
\]

a. KeyError: 'Union'
b. 'Union': 'Union'
c. Union
d. TypeError: ['Bloomfield', 'West Orange']
e. None of the above
Problem 12

def repeatWords(sentence):
    repeats = []
    words = sentence.split()
    for i in range(len(words)-1):
        if words[i] == words[i+1]:
            repeats.append(words[i])
    return repeats

poe = 'Of the bells, bells, bells, bells,'
print(repeatWords(poe))
a. ['bells', 'bells', 'bells']
b. ['bells', 'bells', 'bells', 'bells']
a. ['bells,', 'bells,', 'bells,']
b. ['bells,', 'bells,', 'bells,', 'bells,']
e. None of the above

Problem 13

The lines below are the content of the file named ‘myCat.txt’. After the execution of the following code, what is the content of the file ‘myDog.txt’?

My cat is catatonic
Caterwauling never enters her head

```
inF = open('myCat.txt', 'r')
outF = open('myDog.txt', 'w')
for line in inF:
    line = line.replace('cat', 'dog')
    outF.write(line)
inF.close()
outF.close()
```

a. My dog is dogatonic
   Caterwauling never enters her head
b. My dog is catatonic
   Caterwauling never enters her head
c. My cat is catatonic
   Caterwauling never enters her head
d. My dog is dogatonic
doegerwauling never enters her head
e. none of the above
Problem 14

```
from turtle import *
t = Turtle()
for i in range(4):
    if i % 2 == 0:
        t.up()
        t.forward(100)
        t.right(90)
    if i % 3 == 0:
        t.down()
        t.forward(100)
        t.right(90)
```

a. SyntaxError: invalid syntax
b. One line
c. Two parallel lines
d. Three sides of a square
e. None of the above
The next three problems are short programming problems. (20 points each)

Problem 15.
Write a function named blockWall() that uses turtle graphics to draw a wall composed of square blocks. Each layer of the wall is one block shorter than the one below and is centered on it. blockWall() takes two parameters: len, n, where:

Input:
len is the length of a side of a block
n is the number of blocks in the lowest layer (and also the height of the wall)

For example, the function call blockWall(30, 4) should produce the graphical output below.

```
    
   
  
 

```

The turtle may be left in any position or orientation when blockWall() terminates. For full credit, you must use loops for repeated operations. Hint: write a ‘helper’ function that draws a single block.

Problem 16

Write a function named wordLengths().
Input:
s, a string parameter
Return:
a dictionary in which each key is the length of a word in s and the corresponding value is a list of all distinct words in s of that length

Even if a word occurs more than once in s, it should appear in a value list only once. Treat the text as case sensitive (e.g., ‘August’ and ‘august’ are two different words). Assume, for simplicity, that there is no punctuation in s. Below is correct sample output from wordLengths().

text = 'New York is hardly new'
print(wordLengths(text))

Problem 17.
Write a function named duplicateWordLines() that takes two parameters:

Input:
- inFile -- the name of an existing file to be read
- outFile -- the name of a file into which to write program output

duplicateWordLines() reads the content of inFile, line by line, and writes into outFile any line that contains at least one word greater than three letters long that occurs more than once on the line. The lines in inFile contain only lower case letters and spaces – no capital letters and no punctuation marks.

For example, suppost that inFile contains these two lines

while i nodded nearly napping suddenly there came a tapping
as of some one gently rapping rapping at my chamber door

In this case, only the second line should be written to outFile. (Hint: a line should be written to outFile at most once, no matter how how many times a word occurs or how many words occur more than once.)