CIS 210 Fall 2019 Final Exam

(1-10) Given the following Python code:

```
def parity(bitrep):
    '''(??-1) -> ??-2  '''

    p = 0

    for bit in bitrep:
        if bit == '1':
            p += 1

    if p % 2 == 0:
        p = '0'
    else:
        p = '1'

    return p
```

```
>>> parity('1100011')
```

1) Complete the type contract (replace **??-1** with the appropriate type):

a) `int`   b) `float`   c) `str`   d) `bool`   e) `list`

2) Complete the type contract (replace **??-2** with the appropriate type):

a) `int`   b) `float`   c) `str`   d) `bool`   e) `list`

3) The identifiers `bitrep`, `bit`, and `p` refer to Python objects in a **??** namespace.

a) `local`   b) `global`   c) `builtins`   d) `module`   e) `class`

4) Identifier `parity` refers to Python object of type **??** in a **??** namespace.

a) `class/local`   b) `class/global`   c) `function/local`
d) `function/global`   e) `parameter/local`

5) The first time the `for` loop executes, the value of `bit` is

a) `'1100011'`   b) `True`   c) `False`   d) `'1'`   e) `'0'`

6) The last time the `for` loop executes, the value of `bit` is

a) `'1100011'`   b) `True`   c) `False`   d) `'1'`   e) `'0'`

7) A string method that could do the work of the code in lines 4 through 9 is

a) `startswith` b) `find` c) `index` d) `count` **e)** `split`

8) In line 10, the expression `p % 2` evaluates to

a) `2` b) `1` c) `0` d) `True` e) `False`

9) In line 10, the expression `p % 2 == 0` evaluates to

a) `2` b) `1` c) `0` d) `True` e) `False`

10) At line 15, the value of `p` is

a) `'1100011'` b) `'0'` c) `'1'` d) `0` e) `1`

(11-12) Given the following UNTESTED Python code:

```
def mycharct(s, c):
    '''(str) -> int

    >>> mycharct('x', 'x')          #test 1
    1
    >>> mycharct('x', 'y')          #test 2
    0
    >>> mycharct('xyz', 'x')        #test 3
    1
    >>> mycharct('xyz', 'z')        #test 4
    1
    >>> mycharct('xx', 'x')         #test 5
    2
    >>> mycharct('xaxbx', 'x')      #test 6
    3
    '''
    cct = 0
    for char in s:
        if char == c:
            cct += 1
        return cct
```

11) The tests that reveal the bug are:

a) 1, 2 b) 2, 3, 4 c) 3, 4, 6 d) 4, 5, 6 e) 3, 4, 5, 6

12) If the second line of code were changed to `for char in t`, what type of error message would we see when `mycharct('x', 'x')` was executed?

a) TypeError b) NameError c) ZeroDivisionError d) IndexError e) FileNotFoundError

(13-15) Given the following Python code:

```
>>> from math import pi
>>> dir()
```

13) Which of the following would you see in the global namespace when `dir()` is executed?

a) `import` b) `math` c) `sqrt` d) `math.pi` e) `pi`

14) `math` refers to a Python

a) standard library module b) local namespace c) built-in function d) keyword

15) `dir` will be found in Python

a) standard library module b) local namespace c) builtins d) keyword

16) IDLE is an example of a(n)

a) Python module b) Monte Carlo algorithm c) integrated development environment

17) In the following Python code:

```
>>> r = 9 + 1
>>> s = 'a' + 'b'
```

`+` is an example of a(n) **??** operator.

a) dynamic b) static c) strong d) weak e) overloaded

(18-19) Given the following Python code:

```
def q18 (n, h):
    '''(str, str) -> ??'''
    ctr = 0
    nlen = len(n)
    for i in range(len(h)):
        if h[i:i+nlen] == n:
            ctr += 1
    return ctr
```

18) Complete the type contract:

a) `int` b) `float` c) `str` d) `bool` e) `list`

19) What is the result when the following code is executed?

```
>>> q18('sses', 'assesses')
```

a) `'sses'` b) `'ssesses'` c) `0` d) `1` e) `2`

20) Given the following Python code:

```
def q20r(seq, n):
    ''' (sequence, item) -> boolean'''

    if len(seq) == 0:
        return False
    else:
        mid = len(seq) // 2

        if seq[mid] == n:
            return True
        elif seq[mid] > n:
            return(q20r(seq[:mid], n))
        else:
            return(q20r(seq[mid+1:], n))

>>> q20r((1, 2, 3, 3, 4), 4)
```

The *second* time `q20r` is called, the value of `seq` will be

a) `(1, 2, 3, 3, 4)` b) `(1, 2, 3)` c) `(1, 2)` d) `(3, 4)` e) `''`

(21-22) What will be the result when the following UNTESTED Python code is executed:

```
universities = ['UO', 'OSU', 'WOU']
u1 = universities.pop()
print(universities)         # checkpoint 1
u2 = 'SOU'
universities = universities.append(u2)
print(universities)         # checkpoint 2
```

21) What is printed at checkpoint 1?

a) `['UO', 'OSU', 'WOU']` b) `['UO', 'OSU']`

c) `['OSU', 'WOU']` c) `'WOU'` e) `None`

22) What is printed at checkpoint 2?

a) `['UO', 'OSU', 'SOU']` b) `['UO', 'OSU', 'WOU', 'SOU']`

c) `['SOU', 'UO', 'OSU', 'WOU']` `d) 'SOU'` e) `None`

(23-27) Given the following Python code:

```
1  def q23(astr):
2      '''(str) -> (list of str)'''
3      countd = {}
4      for item in astr:
5          if item in countd:
6              countd[item] += 1
7          else:
8              countd[item] = 1
9
10     countli = countd.values()
11     ct = max(countli)
12
13     mli = []
14     for item in countd:
15         if countd[item] == ct:
16             mli.append(item)
17
18     return mli
```

```
teststr = 'aabbbc'
```

When `q23(teststr)` is executed

23) The value of `countd` at line 9 is

a) `{'b': 3}` b) `[2, 3, 1]` c) `{'a', 'b', 'c'}` d) `['b']`

e) `{'a': 2, 'b': 3, 'c': 1}`

24) The value of ct after line 11 is executed is

a) **1** b) `2` **c) 3** d) `[2, 3, 1]` e) `['b']`

25) The value of `mli` at line 18 is

a) `{'b': 3}` b) `[2, 3, 1]` c) `['a', 'b', 'c']` d) `['b']`

e) `{'a': 2, 'b': 3, 'c': 1}`

26) Which line of code could substitute for lines 13-16?

a) `mli = [item for item in countd if countd[item] == ct]`

b) `mli = [item for item in countd if countd[item] == astr]`

c) `mli = [item[0] for item in countd if countd[item] == ct]`

d) `mli = [item[0] for item in countd]`

e) `mli = [item for item in countd]`

27) Lines 13-16 are an example of what kind of pattern?

a) accumulator b) filter c) map d) Monte Carlo e) binary

(28-30) After the following Python code is executed:

```
course1 = 'CIS 210'
course2 = course1
course1 = course1[:len(course1)-1] + '1'
print(id(course1) == id(course2))           # checkpoint 1
print(course1 == course2)                   # checkpoint 2
```

28) `id(course1)` and `id(course2)` refer to

a) keywords b) modules c) memory locations d) error messages

29) What will be printed at checkpoint 1?

a) `True` b) `False` c) `None` d) error message

30) What will be printed at checkpoint 2?

a) `True` b) `False` c) `None` d) error message

(31-32) After the following Python code is executed:

```
course1 = [2, 1, 0]
course2 = course1
course1[2] = 1
print(id(course1) == id(course2))           # checkpoint 1
print(course1 == course2)                   # checkpoint 2
```

31) What will be printed at checkpoint 1?

a) `True` b) `False` c) `None` d) error message

32) What will be printed at checkpoint 2?

a) `True` b) `False` c) `None` d) error message

33) Given the following UNTESTED Python code:

```
def findRange(salesli):
    '''(list) -> tuple'''

    salesli.sort()
    low = salesli[0]
    high = salesli[-1]
    return low, high

def salesReport(salesli):
    '''(list) --> None

    Prints report of sales totals for each day of week and
    range of per-day sales for the week.

    >>> salesReport([4, 2, 3, 1, 2]
    Weekly Range: $100 - $400

    Mon          Tue          Wed          Thu          Fri
    $400         $200         $300         $100         $200
    '''
    #calculate and report low and high sales
    low, high = findRange(salesli)
    print(f'Weekly Range: ${low * 100} - ${high * 100}\n')

    #print daily report header
    fw = 12
    print(f"{'Mon':<{fw}} {'Tue':<{fw}} {'Wed':<{fw}} \
{'Thu':<{fw}} {'Fri':<{fw}}")

    #report on sales per day from list data
    for sales in salesli:
        print(f'${(sales * 100):<{fw}}', end='')

    return None
```

When the following code is executed, what value is reported for Wednesday?

```
>>> salesReport([4, 2, 3, 1, 2])
```

a) `$100`   b) `$200`   c) `$300`   d) `None`   e) `error`

SA1) Create Python identifiers `name` and `sid`, both of type `str`, and assign them to your name and student id number:

SA2) Replace the ?? with the result of executing the following code (indicate 'error' if the result would be a Python error message):

```
>>> phrase = 'The quick, crafty, hungry brown fox'
>>> phrase = phrase.split(',')
>>> new = phrase[0][5] + phrase[2][-2]
>>> new
??
```

SA3) Write the result when the following Python code is executed:

```
>>> __name__
??
```

SA4) Finish the docstring: supply the ~~type contract and the~~ result of executing `qsa4('sample.txt', '#')` for the following code:

File `sample.txt:`

```
Header1
Header2
Header3
44 55 66
77 88 99
#This line signals end of data.
#This line is a footer.
```

```
def qsa4(f, c):
    '''(str, str)-> None

    Print each line of file f that begins with c.
    Ignore 3 header lines. Remove whitespace characters
    from the end of line before printing.  None is returned.

    >>> qx('sample.txt', '#')
    ??


    '''
    with open(f, 'r') as myf:
        for i in range(3):  #move past header lines
            myf.readline()

        for nextline in myf:
            nextline = nextline.strip()
            if nextline[0] == c:
                print(nextline)
    return None
```

SA5-8) Given the following Python code:

```
def winter(greeting):
    '''(str) → ?? '''

    newgr = ''
    uc = True

    words = greeting.split()
    for word in words:
        if uc:
            word = word.upper()
            uc = False
        else:
            word = word.lower()
            uc = True

        newgr += word + ' '

    return newgr.strip()

>>> greeting = 'Best wishes for a pleasant winter break.'
>>> result = winter(greeting)
```

SA5) Complete the type contract: **??**

SA6) Give the names of the user-defined identifiers that will be in the *global* frame (namespace) *after* the code has been executed: **??**

SA7) Give the names of the user-defined identifiers that will be added to the *local* frame (namespace) while `winter` is executing: **??**

SA8) What is the result when the following code is executed?

```
>>> winter('Best wishes for a pleasant winter break.')
```

a)`'BEST wishes FOR a PLEASANT winter BREAK.'`

b) `BEST wishes FOR a PLEASANT winter BREAK.`

c) `'Best wishes for a pleasant winter break.'`

d) `'Best WISHES for A pleasant WINTER break.'`

e) `'BEST WISHES FOR A PLEASANT WINTER BREAK.'`

SA9) Complete the code for the k-means cluster analysis `createCentroids` function (8 edits):

```
import random

def createCentroids(k, datadict):
   '''(??, dict) -> list (of dict values)

   Create a starter list of k centroids
   for a k-cluster algorithm by
   randomly choosing from the items
   in datadict.  The starter list is returned.

   >>> createCentroids(2,{1: [2.77], 2: [2.97], 3: [4.05]})
   [[2.77], [2.97]]
   '''
   centroids = ??
   centroidCount = ??
   centroidKeys = ??

   while centroidCount < ??:
      rkey = random.randint(1, len(datadict))

      #keys cannot repeat
      if ?? not in centroidKeys:
         centroids.append(??)
         centroidKeys.append(rkey)
         centroidCount += 1

   return ??
```

NAME____ STUDENT ID________________

Write a function, findLast, with parameters s, a string, and, c, a string of length 1. findLast will return the index (index >= 0) of the last occurrence of c in s. If c does not occur in s, findLast should return −1.

a) Write a simple example of use for findLast:

b) Write one edge test case for findLast – note WHY it is an edge case:

c) Write the function header for findLast:

d) Write the type contract for findLast:

e) Write the rest (i.e., do not include header and docstring) of the Python code for findLast: