CIS 210 Fall 2019 Midterm Exam

(1) Computational Problem Solving - Replace ?? with the correct response.

TASK/PROBLEM → Computational Thinking ↔
SOLUTION/ALGORITHM/COMPUTATIONAL PROCESS

ALGORITHM ↔ Design/Coding ↔
COMPUTER PROGRAM

COMPUTER PROGRAM ↔ ?? ↔
HIGH QUALITY (RELIABLE/REUSABLE) COMPUTER PROGRAM

a) looping   b) executing   c) printing   d) testing and debugging

(2-3) Given the following Python code:

def charCt(s, c):
    '''(str, str) -> int
    Return count of occurrences of char c in string s.'''
    ct = 0
    for ch in s:
        if ch == c:
            ct += 1
    return ct

(2) The set of test cases that will NOT find the bug in charCt is

a) charCt('','a')   b) charCt('abc','a')   c) charCt('abc','a')
charCt('a','a')     charCt('abc','b')     charCt('def','b')
charCt('hi','o')    charCt('abc','c')     charCt('ghi','x')
charCt('hello, world','o')  charCt('x','x')

(3) The bug in charCt is an example of which kind of error?

a) syntax   b) runtime   c) logical   d) NameError   e) TypeError
The Taylor series expansion for $e^x$ has the following form:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

If we truncate the sum after $N$ terms, we have an approximation for $e^x$ as:

$$e^x \approx \sum_{n=0}^{N-1} \frac{x^n}{n!}$$

Given the following Python code, consistent with this approximation for $e^x$:

```python
from math import factorial  # factorial(n) returns n!

def approx_e(x, i):
    '''(number, int) -> ??-1

    Generate approximation for e**x with i iterations of the
    Taylor series expansion for e**x. The approximate value of
    e**x is returned.
    
    >>> approx_e(1, 100)
    2.7182818284590455
    ...'''
    acc = 0
    for n in range(??-2):
        acc += pow(x, n)/factorial(n)
    return acc
```

(4) Complete the docstring (??-1):

a) float  
   b) int  
   c) number (int or float)  
   d) bool  
   e) str

(5) Complete the call to Python `range` (??-2):

a) i  
   b) i - 1  
   c) i + 1  
   d) x  
   e) x - 1

(6) `approx_e` is defined in what type of namespace?

a) local  
   b) global  
   c) built-in  
   d) both local and global
(7) What is the result of executing \(q7(75)\)?

```
def q7(temp):
    ''' midterm function '''
    warning = ''

    if temp >= 90:
        warning = 'very hot'

    if temp >= 80:
        warning = 'hot'

    if temp >= 70:
        warning = 'ok'

    if temp >= 60:
        warning = 'cool'

    return warning

a) 'very hot'  b) 'hot'  c) 'ok'  d) 'cool'  e) ''
```

(8) What is the result of executing \(q8()\)?

```
def q8():
    ''' midterm function '''
    x = 3210
    count = 0
    while x > 0:
        count += 1
        x = x // 10
    return count

a) 0  b) 1  c) 2  d) 3  e) 4
```
(9-10) Given the following UNTESTED Python code, what is the result of executing

```python
def q9(s):
    '''(str) -> int

    Returns the length of the longest single-character string in s.

    >>> q9('abcccdf')  # normal example
    3
    >>> q9('')         # empty string
    0
    ...
    if len(s) != 0:
        prev_char = s[0]
        dup_ct = 1
        high_ct = 1
    else:
        high_ct = 0

    for i in range(1, len(s)):
        if s[i] == prev_char:
            dup_ct += 1
        else:
            prev_char = s[i]
            if dup_ct > high_ct:
                high_ct = dup_ct
            dup_ct = 1

    return high_ct
```

(9) >>> mystr = 'aa'
>>> q9(mystr)?

a) 0   b) 1   c) 2   d) 3   e) NameError

(10) >>> s
a) 'aa'   b) 'a'   c) '2'   d) ''   e) NameError
(11-13) Complete the docstrings for the following Python code:

```python
def hello(s):
    '''(??-1) -> None
    ...
    print('Hello, ' + s + '.')
    return None

def ciao(s):
    '''(str) -> ??-2
    ...
    print('Ciao, ' + s + '.')
    return None

def greeting(f, s):
    '''(??-3, str) -> None
    ...
    f(s)
    return None
```

11) Replace ??-1
   a) int       b) bool     c) str       d) function   e) None

12) Replace ??-2
   a) int       b) bool     c) str       d) function   e) None

13) Replace ??-3
   a) int       b) bool     c) str       d) function   e) None

14) Python will locate the print function used in ciao in the ?? namespace
    a) local     b) global    c) __main__     d) __builtins__

15) A Python function that implements the fizzbuzz game for a positive integer should use
    a(n)
    a) for loop   b) while loop c) import statement d) Monte Carlo method
16) In Python assignment
   a) the expression on the right hand side of the assignment operator is evaluated
   b) a Python namespace may be affected       c) no value is returned
   d) all of (a)-(c)       e) some of (a)-(c)

17) When >>> def = 123 and >>> abs = 456 are executed, the result is
   a) new value for def / new value for abs       b) new value for def / error
   c) error / new value for abs       d) error / error

18) Python functions `__??__` return a value and `__??__` cause a side effect.
   a) always/never       b) always/sometimes       c) always/always
   d) sometimes/always       e) sometimes/sometimes

19) What is the result of executing the following Python code:

    def twice(x):
        '''midterm function '''
        result = 2 * x
        #print(result)
        return None
    
    def thrice(x):
        '''midterm function '''
        result = twice(x) + x
        return result
    
    >>> thrice(5)
    ??

   a) 5       b) 10       c) 10       d) TypeError       e) NameError

20) What is the result of executing the following Python code:

    >>> (isinstance(99, int) == True) == (isinstance(99, int))
   a) 99       b) True       c) False       d) None       e) TypeError
(21-23) Given the following Python code:

```python
1 def midxq(s):
2     '''(str) -> str
3     Exam function.
4     '''
5     new_s = ''
6     for ch in s:
7         new_s = ch + new_s
8     return new_s
9
10 name = 'Tuesday'
11 new_name = midxq(name)
```

(21) Give the names of the user-defined identifiers that will be in the local frame (namespace) when the code is executed (line 7 of `midxq`):

(22) Give the names of the user-defined identifiers that will be in the global frame (namespace) after the following code is executed:

(23) Write the value of `new_name` after the code is executed: __________

(24-31) Write the result when the following Python code is executed:

```python
>>> r1 = __name__
>>> r1
??__________________
>>> type(r1)
??__________________

>>> r2 = round(6.5)
>>> r2
??__________________
>>> type(r2)
??__________________

>>> r3 = 'CIS 210'.find('2')
>>> r3
??__________________
>>> type(r3)
??__________________

>>> r4 = r2 != 0
>>> r4
??__________________
>>> type(r4)
??__________________
```
(32) Given the following Python code:

```
1  def drawShape(s):
2      '''(int) --> None
3      Draw a square with sides of length s.
4      >>> drawShape(100)
5      [draws a square with sides length 100]
6      '''
7      turn = 90
8      for i in range(4):
9          fd(s)
10         lt(turn)
11      return None
```

Revise (mark up this code) `drawShape` to draw an n-sided polygon, where `n` is a new argument to the function (7 places/lines of code).
(33) Background - Fizzbuzz

Recall that fizzbuzz is played as follows: starting at 1, count up to some number, \( n \). For each number, if it is divisible by 3, say (display) “fizz”. If it is divisible by 5, display “buzz”. If it is divisible by both 3 and 5, display “fizzbuzz”. Otherwise, just display the number.

Requirements

Write a Python function, \( fb \), that implements fizzbuzz. Function \( fb \) should have one parameter, \( n \), which is the number that will end the game.

For each round of fizzbuzz, the function should print the number or “fizz” or “buzz” or “fizzbuzz”. At the end of the game, print “Game over!” \( fb \) should return the None value.

For example,

```python
>>> fb(16)
1
2
fizz
4
buzz
fizz
7
8
fizz
buzz
11
fizz
13
14
fizzbuzz
16
Game over!
```

Your code should be written using CIS 210 style guidelines:

- include a docstring (type contract ONLY needed here)
- use whitespace between operators and operands
- use descriptive variable names
- add appropriate comments

(You may omit the program header.)