THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

This exam paper must not be removed from the venue

Venue _____

Seat Number _____

Student Number |__|__|__|__|__|__|__|__|

Family Name _____

First Name _____

School of Information Technology and Electrical Engineering

EXAMINATION

Semester One Final Examinations, 2019

CSSE1001 / CSSE7030 Introduction to Software Engineering

This paper is for St Lucia Campus students.

Examination Duration: 120 minutes

Reading Time: 10 minutes

Exam Conditions:

This is a Central Examination

This is a Closed Book Examination - no materials permitted

During reading time - write only on the rough paper provided

This examination paper will be released to the Library

Materials Permitted In The Exam Venue:

(No electronic aids are permitted e.g. laptops, phones)

Calculators - No calculators permitted

Materials To Be Supplied To Students:

1 x Multiple Choice Answer Sheet

Instructions To Students:

Additional exam materials (eg. answer booklets, rough paper) will be provided upon request.

For Examiner Use Only

Question	Mark

Total _____

For all questions, please choose the most appropriate answer if it appears that more than one option is a potentially correct answer. All coding questions relate to the Python 3 programming language. If an evaluation produces an error of any kind, choose Error as your answer. Different questions may have different numbers of choices. Each question is worth one mark.

1. What does the expression `4 + 7 // 2` evaluate to?

   a) `5`

   b) `5.5`

   c) `7`

   d) `7.5`

   e) `Error`

2. What does the expression `[1, 2] + [2, 3]` evaluate to?

   a) `[1, 2, 2, 3]`

   b) `[1, 2, 3]`

   c) `[3, 5]`

   d) `Error`

3. What does the expression `(1, 'a') + (2, 'b')` evaluate to?

   a) `{1: 'a', 2: 'b'}`

   b) `(1, 'a', 2, 'b')`

   c) `(3, 'ab')`

   d) `[(1, 'a'), (2, 'b')]`

   e) `Error`

4. What does the expression `(0 < 2 < 4 and not 3 > 0 > 1)` evaluate to?

   a) `0`

   b) `4`

   c) `True`

   d) `False`

   e) `Error`

5. After the assignment `s1 = "Programming" + " is " + "Fun"`, which of the following statements assigns `"is"` to `s2`?

   a) `s2 = s1[1]`

   b) `s2 = s1[-6:-4]`

   c) `s2 = s1[12:13]`

   d) `s2 = s1[12:14]`

   e) More than one of the above is correct.

6. What is the value of `s4` after the following statements are evaluated?

```
s1 = "Hello"
s2 = "World"
s3 = "Ni Hao"
s3 = s1
s3 = "Hi"
s4 = s1 + s2
```

a) `"HiWorld"`

b) `"HelloWorld"`

c) `"Ni HaoWorld"`

d) `Error`

7. What is the value of `a` after the following statements are evaluated?

```
x = ['x', 'y', 'z']
y = ['z', 'y', 'x']
z = x + y
a = z[1]
```

a) `'x'`

b) `'y'`

c) `['x', 'y', 'z']`

d) `['z', 'y', 'x']`

e) `Error`

8. What is the value of `x` after the following statements are evaluated?

```
x = [-1, 1.5, 'a']
y = x
y[2] = 0
```

a) `[-1, 1.5, 0]`

b) `[-1, 0, 'a']`

c) `-1`

d) `0`

e) `Error`

9. What is the value of `x` after the following statements are evaluated?

```
x = [1, 2, 3]
x.pop(1)
y = x.extend([4, 5, 6])
x.pop(3)
```

a) `[1, 2]`

b) `[5, 6]`

c) `[2, 4, 5, 6]`

d) `[1, 3, 4, 6]`

e) `Error`

10. Why can a list not be used as a key for a value in a dictionary? i.e. The following code will raise an error:

```
d = {[1,2]:'a', [4,5]:'b'}
```

a) The Python interpreter cannot determine which list value to use as the key.

b) A list may be empty, so its contents cannot be hashed to generate an index.

c) Dictionary keys must be immutable types, so that the key's index does not change.

d) Lists may contain elements of different types and all keys in a dictionary must be of the same type.

11. What is the value of `d` after the following statements are evaluated?

```
d = {'eng':'Hello', 'fre':'Bonjour',
     'spa':'Hola',  'chi':'Ni Hao'}
d['spa'] = 'Buenos Dias'
d.get('jpn', 'Konichiwa')
```

a) `{'spa': 'Buenos Dias'}`

b) `{'eng':'Hello', 'fre':'Bonjour',`
`'spa':'Hola',  'chi':'Ni Hao'}`

c) `{'eng': 'Hello', 'fre': 'Bonjour',`
`'spa': 'Buenos Dias', 'chi': 'Ni Hao'}`

d) `{'eng': 'Hello', 'fre': 'Bonjour', 'spa': 'Buenos Dias',`
`'chi': 'Ni Hao', 'jpn': 'Konichiwa'}`

e) `Error`

12. What is the value of the global variable a after the following code is executed?

```
def f(x) :
    a = 3
    x = x / a
    return (a + x) % x

a = 9
f(a)
```

a) 9

b) 3

c) 0

d) 0.0

e) Error

13. What is the value of x after the following code is executed?

```
def f1(a, b) :
    return a / b

def f2(a, b) :
    b = 10
    return f1(b, a)

x = f2(8, 4)
```

a) 0.5

b) 1.25

c) 2.0

d) 2.5

e) Error

14. What is output after the following code is executed?

```
x = 0
stars = '*'
while x > 0 :
    print(stars)
    stars += '*'
    x -= 1
```

a) *

b) A blank line

c) There is no output

d) An infinite number of *'s

e) Error

15. What is output after the following code is executed?

```
x = 100
if x > 1 :
    print("positive")
elif x > 10 :
    print("large positive")
elif x == 0 :
    print("zero")
else :
    print("negative")
```

a) positive

b) large positive

c) positive
large positive

d) negative

e) Error

16. What is output after the following code is executed?

```
daily_high_temperature = [29, 30, 27, 32, 31, 28, 33]
yesterday_temperature = 30
hotter = 0
colder = 0

for temperature in daily_high_temperature :
    if yesterday_temperature < temperature :
        hotter += 1
    elif yesterday_temperature > temperature :
        colder += 1

if hotter > colder :
    print("getting hotter")
elif colder > hotter :
    print("getting colder")
else :
    print("no change")
```

a) getting hotter

b) getting colder

c) no change

d) There is no output

e) Error

17. Assuming that the parameter `lst` is a list of integer values, which of the following descriptions best describe the purpose of this function?

```
def f(lst) :
    a = 0
    b = 0
    for x in lst :
        if x % 2 == 0 :
            a += 1
        else :
            b += 1
    return a, b
```

a) It returns the average of all even integers and all odd integers in the list.

b) It returns the sum of all even integers and all odd integers in the list.

c) It returns how many even and odd integers were in the list.

d) It raises an error as a function cannot return more than one value.

18. Why are global constants considered to be good programming style, while global variables are bad programming style?

a) Global constants are still global values and are bad programming style and should be avoided.

b) In the Python interpreter, as global constants do not change value, they can be cached in high speed memory to provide faster lookup times for their values.

c) Constants do not change value, so they allow meaningful names to be given to commonly used values that can be referenced anywhere in a program.

d) Both (b) and (c).

19. What is the purpose of the `try` statement in Python?

a) To indicate that the code has encountered an error it cannot handle locally.

b) To provide an error handling function that will be called if any error occurs in a block of code.

c) To guarantee that any errors raised by a block of code will be handled, so that the program will not crash.

d) To attempt to execute a block of code and handle at least some of the errors that may be raised by the statements in the block of code.

20. What is output after the following code is executed:

```
def f2(a, b) :
    if b == 0:
        raise ValueError("Cannot be '0'.")
    return a / b

def f1(a, b) :
    return f2(a, b)

x = 1
y = 0
try :
    print("Answer is", f1(x, y))
except ValueError as e:
    print(str(e))
```

a) `Answer is 0.0`

b) `Answer is 1.0`

c) `Cannot be '0'.`

d) There is no output because the `ZeroDivisionError` was not handled by an `except` clause.

e) There is no output as the program crashes because the error was not handled in function `f1`.

21. For the following block of code:

```
options = {'a': "Option A",
           'b': "Option B",
           'c': "Option C",
           'd': "Option D",
           'e': "Option D"}
print('a)', options['a'])
print('b)', options['b'])
print('c)', options['c'])
print('d)', options['d'])
print('e)', options['e'])
```

Which of the following programming constructs would **best** simplify the above code?

a) for loop

b) while loop

c) if statement

d) function

e) class

22. For the following block of code:

```
daily_high_temperature = [29, 30, 27, 32, 31, 28, 33]
daily_low_temperature = [17, 20, 15, 19, 21, 17, 19]

total = 0
for temp in daily_high_temperature :
    total += temp
print("The average high temperature was",
      total / len(daily_high_temperature))

total = 0
for temp in daily_low_temperature :
    total += temp
print("The average low temperature was",
      total / len(daily_low_temperature))
```

Which of the following programming constructs would **best** simplify the above code?

a) tuple
b) dictionary
c) if statement
d) while loop
e) function

23. For the following function:

```
def r(x) :
    if x == 0 :
        return x
    else :
        return r(x-1) + x
```

What will `r(5)` return?

a) `0`
b) `10`
c) `15`
d) `21`
e) `Error`

24. The following is a recursive function to check to see if an `item` is in a list of numbers. It assumes that the values in `alist` are stored in ascending order.

```
def in_list(alist, item) :
    if len(alist) == 0 :
        return False
    else :
        midpoint = len(alist) // 2
        if alist[midpoint] == item :
            return ## Fragment 1
        else :
            if item < alist[midpoint] :
                return ## Fragment 2
            else :
                return ## Fragment 3
```

Example usage:

```
in_list([1, 4, 9, 13, 25], 13) == True
in_list([1, 4, 9, 13, 25], 17) == False
```

Which code fragments will correctly complete the function above?

a) Fragment 1 is: `True`
Fragment 2 is: `in_list(alist[:midpoint], item)`
Fragment 3 is: `in_list(alist[midpoint+1:], item)`

b) Fragment 1 is: `True`
Fragment 2 is: `in_list(alist[midpoint+1:], item)`
Fragment 3 is: `in_list(alist[:midpoint], item)`

c) Fragment 1 is: `False`
Fragment 2 is: `in_list(alist[:midpoint], item)`
Fragment 3 is: `in_list(alist[midpoint+1:], item)`

d) Fragment 1 is: `False`
Fragment 2 is: `in_list(alist[midpoint+1:], item)`
Fragment 3 is: `in_list(alist[:midpoint], item)`

e) None of the code fragments would implement the function correctly.

The next three questions refer to the following function definition, which is missing three lines of code. The function reads raw wind speed data from a file and calculates the average and maximum wind speeds for each day. The following is an example of a data file (wind_speed.csv).

```
12,15,8,24,2,15
22,12,19,29,16,13
11,5,7,3,5,9,2,4,1,3,5,7,12,8,11
```

Each line of the file contains the wind speed readings collected on a single day. These values may be integer or floating point and are separated by a comma. Each day may have different numbers of readings. The average wind speed for a day is simply the average of all the values on the line. The maximum wind speed for a day is the largest value on the line. The results are written to an output file in the same order in which they are read from the input file. The logic assumes that the data in the input file is in the correct format.

The definition of the `process` function, with three missing lines, is given below.

```
def process(wind_speeds, processed_data) :
    with open(wind_speeds, "r") as wind_data, \
         open(processed_data, "w") as result_data :
        for day in wind_data :
            day = day.strip()
            wind_speeds = day.split(',')
            ## line 1: Set initial max wind speed ##
            total = 0
            ## line 2: Calculate average and max wind speeds ##
            ## line 3: Update max wind speed ##
            average_wind_speed = total / len(wind_speeds)
            result_data.write(f"{str(average_wind_speed)}," +
                              f"{str(max_wind_speed)}\n")
```

The result of calling the completed function on the file described above, for example by:

```
process('wind_speed.csv', 'processed_results.csv')
```

Would result in the following data being saved to processed_results.csv.

```
12.666666666666666,24.0
18.5,29.0
6.2,12.0
```

When answering questions 25 and 27, assume that the correct code has been implemented from the previous question(s).

25. What code is required at **## line 1: Set initial max wind speed. ##**?

a)
```
max_wind_speed = wind_speeds
```

b)
```
max_wind_speed = wind_speeds[0]
```

c)
```
max_wind_speed = float(wind_speeds)
```

d)
```
max_wind_speed = float(wind_speeds[0])
```

26. What code is required at **## line 2: Calculate average and max wind speeds ##**?

a)
```
for wind_speed in day :
    wind_speed = float(wind_speed)
    total += wind_speed
```

b)
```
for wind_speed in day :
    total += wind_speed
```

c)
```
for wind_speed in wind_speeds :
    wind_speed = float(wind_speed)
    total += wind_speed
```

d)
```
for wind_speed in wind_speeds :
    total += wind_speed
```

27. What code is required at **## line 3: Update max wind speed ##**?

a)
```
if wind_speed > max_wind_speed :
    max_wind_speed = wind_speed
```

b)
```
if wind_speeds[0] > max_wind_speed :
    max_wind_speed = wind_speeds[0]
```

c)
```
if float(wind_speed) > float(max_wind_speed) :
    float(max_wind_speed) = float(wind_speed)
```

d)
```
max_wind_speed = max(wind_speed)
```

The following partial definition of a student class is used in the following three questions.

```
class Student(object) :
    def __init__(self, student_number, name) :
        """
        Parameters:
            student_number (str): Student's id number.
            name (str): Student's full name.
        """
        self._student_number = student_number
        self._name = name
        self._results = { } # Dictionary of results,
                            # course code is the key
                            # mapped to grade achieved.

    def add_result(self, course_code, grade) :
        """Record grade student achieved in course_code.

        Parameters:
            course_code (str): Course student has completed.
            grade (int): Grade student achieved (0-7).

        Pre-conditions:
            0 <= grade <= 7
        """
        ## code block 1 ##

    def get_gpa(self) :
        """(int) Return the student's current GPA.
           GPA is recalculated each time this method is called.
        """
        ## code block 2 ##
```

28. What is the required code for **`## code block 1 ##`**?

a) `self._results[grade] = course_code`

b) `self._results[course_code] = grade`

c) `self._results.get(course_code, grade)`

d) `self._results.set(course_code, grade)`

e) None of the statements above are correct.

29. What is the required code for **`## code block 2 ##`**?

a)
```
return sum(self._results) / len(self._results)
```

b)
```
sum_of_grades = 0
for grade in self._results :
    sum_of_grades += grade
return sum_of_grades / len(self._results)
```

c)
```
sum_of_grades = 0
for course in self._results :
    sum_of_grades += self._results.pop(course)
return sum_of_grades / len(self._results)
```

d)
```
sum_of_grades = 0
for course, grade in self._results.items() :
    sum_of_grades += grade
return sum_of_grades / len(self._results)
```

e) None of the statements above are correct.

30. After the following statement is executed:

```
me = Student("12345678", "My Name")
```

Which of the following sets of statements will result in `"GPA is: 6.0"` being output?

Assume that **`## code block 1 ##`** and **`## code block 2 ##`** contain the correct code.

a)
```
Student.add_result(me, "CSSE1001", 6)
Student.add_result(me, "INFS1200", 6)
print("GPA is:", Student.get_gpa(me))
```

b)
```
Student.add_result("CSSE1001", 6)
Student.add_result("INFS1200", 6)
print("GPA is:", Student.get_gpa())
```

c)
```
me.add_result("CSSE1001", 6)
me.add_result("INFS1200", 6)
print("GPA is:", me.get_gpa())
```

d)
```
add_result(me, "CSSE1001", 6)
add_result(me, "INFS1200", 6)
print("GPA is:", get_gpa(me))
```

e) None of the statements above will result in "GPA is: 6.0" being output.

31. What is an advantage of using classes to structure your program over just using functions?

    a) Classes provide a mechanism to encapsulate data and behaviour together. This means that only the methods defined in the class will manipulate an object's data. Consequently, there is less chance of unknown parts of a program using an object's data and breaking the program's logic.

    b) Classes are a mechanism to group related functions together that reduces the amount of code that needs to be written in each function. This mechanism inherently reduces duplication between similar functions. Consequently, using classes improves your code readability.

    c) Classes provide a mechanism to group related functions into a single file. This simplifies maintenance of the code, as all similar functionality is stored in a single file. Consequently, it is faster to search for the implementation of functionality across files in the file system.

    d) Classes make it easier to reuse code. Whenever a programmer wants to reuse a method, they can inherit from the class that defines the method and then call that method in their code. Consequently, it simplifies access to existing methods.

    e) None of the answers above are valid descriptions of an advantage of using classes to structure your program over just using functions.

The next four questions refer to the following class definitions.

```
class A :
    def __init__(self, x) :
        self._x = x

    def f(self, x) :
        return self.g(x) + 1

    def g(self, x) :
        return x * x

class B(A) :
    def g(self, y) :
        return self._x * y

class C(B) :
    def __init__(self, x, y) :
        super().__init__(x)
        self._y = y

    def f(self, x) :
        return x * self._y

class D(B) :
    def __init__(self, x, y) :
        super().__init__(x)
        self._x -= y
        self._y = y

    def f(self, x) :
        return super().f(x) * x

    def g(self, y)  :
        return self._y * y

a = A(4)
b = B(2)
c = C(4, 3)
d = D(4, 1)
```

32. What does `a.g(2)` return?

a) 2

b) 3

c) 4

d) 16

33. What does `b.f(3)` return?

    a) `7`

    b) `9`

    c) `10`

    d) `13`

34. What does `c.g(2)` return?

    a) `4`

    b) `6`

    c) `8`

    d) `9`

35. What does `d.f(2)` return?

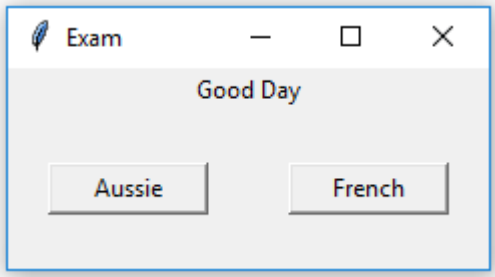
    a) `2`

    b) `6`

    c) `8`

    d) `10`

The next two questions relate to the following simple GUI application. The application has two buttons and a label. The label is initially “Good Day”. When started, the GUI appears as in the image below.



Clicking on the button labelled “Aussie”, changes the label to be “G’Day”. Clicking on the button labelled “French”, changes the label to be “Bonjour”. The code is provided below.

```
import tkinter as tk

class App(object) :

    def __init__(self, master) :
        """Initialise the window layout with a greeting label
           and buttons to change the greeting language.

        Parameters:
            master (Tk): Main window for application.
        """
        master.title("Exam")
        master.geometry('240x100')
        self._label = tk.Label(master, text="Good Day")
        self._label.pack()
        ## code block 1 ##
        ## code block 2 ##
        self._canvas = tk.Canvas(master)
        self._canvas.pack()

    def _set_greeting(self, greeting) :
        self._label.config(text=greeting)

    def _aussie_greeting(self) :
        self._set_greeting("G'Day")

    def _french_greeting(self) :
        self._set_greeting("Bonjour")
```

36. What is the required code for **`## code block 1 ##`**?

a)
```
english_btn = tk.Button(master, text="Aussie", width=10,
                        command=_aussie_greeting())
french_btn = tk.Button(master, text="French", width=10,
                       command=_french_greeting())
```

b)
```
english_btn = tk.Button(master, text="Aussie", width=10,
                        command=_aussie_greeting)
french_btn = tk.Button(master, text="French", width=10,
                       command=_french_greeting)
```

c)
```
english_btn = tk.Button(master, text="Aussie", width=10,
                        command=self._aussie_greeting())
french_btn = tk.Button(master, text="French", width=10,
                       command=self._french_greeting())
```

d)
```
english_btn = tk.Button(master, text="Aussie", width=10,
                        command=self._aussie_greeting)
french_btn = tk.Button(master, text="French", width=10,
                       command=self._french_greeting)
```

e) More than one of the above is correct.

37. What is the required code for **`## code block 2 ##`**?

a)
```
english_btn.pack(side=tk.BOTTOM, padx=20)
french_btn.pack(side=tk.BOTTOM, padx=20)
```

b)
```
english_btn.pack(side=tk.LEFT, padx=20)
french_btn.pack(side=tk.RIGHT, padx=20)
```

c)
```
english_btn.pack(side=tk.BOTTOM)
french_btn.pack(side=tk.BOTTOM)
```

d)
```
english_btn.pack(side=tk.LEFT)
french_btn.pack(side=tk.RIGHT)
```

e) More than one of the above is correct.

38. What is the time complexity, in terms of the length of the list of values, of the following function that returns the index of the largest value in the list? You may assume accessing elements of a list, determining the length of a list and arithmetic operations are all constant time operations.

```
def find_largest(values) :
    """Returns position of the largest number in 'values'.

    Parameters:
        values (list[float]): List of numbers.

    Return:
        (int) Position of the largest number in 'values'.

    Pre-condition:
        values is not an empty list.
    """
    largest_position = 0
    index = 1
    while index < len(values) :
        if values[index] > values[largest_position] :
            largest_position = index
        index += 1
    return largest_position
```

a) O(1) – Constant

b) O(log n) – Logarithmic

c) O(n) – Linear

d) O(n^2) – Quadratic

e) O(2^n) – Exponential

The next two question relate to the following function definition. The function tests to see if the year passed as a parameter is a leap year.

```
def is_leap_year(year) :
    """Check if 'year' is a leap year.

    Parameters:
        year (int): Year to check if it is a leap year.

    Return:
        (bool) True if 'year' is a leap year, False otherwise.

    Pre-condition:
        year > 0
    """
    if year % 400 == 0 :
        return True
    elif year % 100 == 0 :
        return False
    elif year % 4 == 0 :
        return True
    else :
        return False
```

39. What is the time complexity, in terms of the logical complexity of the function above? You may assume arithmetic operations are constant time operations.
    a) O(1) – Constant
    b) O(log n) – Logarithmic
    c) O(n) – Linear
    d) O(n2) – Quadratic
    e) O(2n) – Exponential

40. Assume each of the following sets of values are individually passed as parameters to the `is_leap_year` function. Which of these sets of values, would be the fewest number of inputs that would be required to test all logical paths in the function?
    a) 2018, 2019, 2020.
    b) 1900, 2000, 2019, 2020.
    c) 1600, 1800, 1900, 2000, 2018, 2019, 2020.
    d) -400, -100, -5, -4, 0, 4, 5, 100, 400, a, aa, a4, a-4.
    e) None of the above.

END OF EXAMINATION