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, 2018

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 `(4.4 + 6.6) // 2` evaluate to?

   a) 5

   b) 5.0

   c) 5.5

   d) Error

2. What does the expression `4 + 5 / 2` evaluate to?

   a) 4

   b) 6

   c) 4.5

   d) 6.5

   e) Error

3. What does the expression `(2.0**3) % 3` evaluate to?

   a) 0

   b) 2

   c) 0.0

   d) 2.0

   e) Error

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

   1, 2, -3 --> (1,2,-3)
   1,2,[3] --> (1,2,[3])

   a) 6

   b) (6,)

   c) (1, 2, 3)

   d) Error

5. What does the expression `('a', 'c') + ('b', 'd')` evaluate to?

   a) ('a', 'c', 'b', 'd')

   b) ('a', 'b', 'c', 'd')

   c) ('ab', 'cd')

   d) 'acbd'

   e) Error

6. What is the result of `1 < 2 and not 2 > 3`?

a) 2

b) True

c) False

d) Error

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

```
x = [1, 2, 3]
y = [4, 5, 6]
z = x + y
a = z[4]
```

z=[1,2,3,4,5,6]

a) 4

b) 5

c) [4]

d) [5]

e) Error

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

```
x = ['a', 'b', 'c']
y = x
y[1] = 0
```

assign mutable variable to another variable
if change the assigned variable,
the original variable would change

a) 0

b) ['a']

c) ['a', 0, 'c']

d) ['a', 'b', 'c']

e) Error

9. After the assignment `s1 = "Monty Python"`, which of the following statements assigns `'Py'` to `s2`?

a) `s2 = s1[6:2]`

b) `s2 = s1[-6:-2]`

c) `s2 = s1[6:7]`

d) `s2 = s1[6:8]`

e) More than one of the above is correct.

10. What does the following expression evaluate to?

```
('abc' + 'def' + 'ghi')[2]
```

join the string

a) 'c'

b) 'b'

c) 'ghi'

d) 'def'

e) Error

11. Given the following code:

input is string

```
x = input("Please enter the first number: ")
y = input("Please enter the second number: ")
print ("x + y =", x + y)
```

and assuming that the user inputs 2 and 5 respectively. What would be the output?

a) x + y = x + y

b) x + y = 25

c) x + y = 7

d) Error

read the question carefully

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

```
d = {1:'a', 2:'b', 3:'c'}
d[4] = 'd'
d.get(5, 'e')
```

a) {5:'e'}

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

c) {1:'a', 2:'b', 3:'c', 4:'d'}

d) {1:'a', 2:'b', 3:'c', 4:'d', 5:'e'}

e) Error

13. Consider the following two lines of code.

```
d = {1:'a', 2:'b', 3:'c'}
d[['d']] = 4
```

Which of the following statements best explain why this code raises an error when executed?

a) Dictionary keys must be immutable types.

b) There is no value mapped to the key `['d']`.

c) There is a syntax error in the second statement.

d) All keys in a dictionary must be of the same type.

e) All values in a dictionary must be of the same type.

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

```
def f(x) :
    a = 100
    x += a
    return x + a

a = 1
f(a)
```

a) 1

b) 100

c) 101

d) 201

e) Error

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

```
def f(l, a, b) :
    l.append(a)
    l = l + [b]
    return l

x = [5, 9]
x = f(x, 2, 1) + x
```

x=[5,9]

[5,9,2]

[5,9,2,1]+[5,9,2]

a) [5, 9, 2, 1]

b) [5, 9, 2, 1, 5, 9]

c) [5, 9, 2, 1, 5, 9, 2]

d) [5, 9, 2, 1, 5, 9, 2, 1]

e) Error

16. The syntax of the `for` statement is:

```
for <item> in <items> :
```

Which of the following is true about *<items>*?

a) It can only be a list.

b) It cannot be a tuple.

c) It must be sorted in ascending order.

d) It can be any sequence, like a string or list.

17. Which of the following descriptions best describe the purpose of this function?

```
def f() :
    t = 0
    r = int(input('Please input an integer: '))
    while r != 0 :
        if r % 2 == 0 :
            t += r
        r = int(input('Please input an integer: '))
    return t
```

a) It does not do anything as the body of the while loop never executes.
b) It is an infinite loop as the while loop condition can never be false.
c) It returns the sum of all integers entered.
d) It returns the sum of all odd integers entered.
e) It returns the sum of all even integers entered.

18. Why is it good practice to use constants in your code?

a) They make it easier to modify the code when values change.
b) They allow the same value to be used multiple times without being redeclared as a literal value.
c) They improve the readability of the code by providing a descriptive name for a value rather than just the value.
d) All of the above.
e) None of the above.

19. Which of the following statements best describes the relationship between a class and an object?

a) A class is a programming construct used to describe one or more objects.
b) An object is a variable defined in side of a class.
c) A class is an instance of an object.
d) They are the same thing.

20. For the following block of code:

```
student_numbers = ["s0123456", "s1234567",
                   "s2345678", "s3456789"]
assessment_marks = [[10, 19, 18, 10, 38]
                    [8, 17, 15, 9, 35]
                    [5, 20, 8, 7, 7, 33]
                    [4, 18, 7, 6, 6, 29]]
grades = [7, 6, 6, 5]
course = [CSSE1001, CSSE1001, INFS1200, INFS1200]
```

Which of the following programming constructs would be **best** suited to making the above code more structured and maintainable?

a) a dictionary

b) a function

c) a class

d) a tuple

21. What is the purpose of raising an exception in your code?

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

b) To indicate that this block of code will resolve an error encountered in a previous function call.

c) To indicate that this block of code will attempt to execute some statements that may cause an error.

d) To prevent the Python interpreter from halting execution of a program when an error is encountered. try

22. Consider the following code:

```
class AnException(Exception) :
    pass

def exceptions(l, a) :
    l.append(9)
    l.pop(a)
    raise AnException()

exceptions([1, 2, 3], 4)
```

[1,2,3,9]

Which of the following exceptions, if any, will the code above raise?

a) TypeError

b) IndexError

c) AnException

d) AttributeError

e) No exception will be raised.

The following recursive function definition is used in the next two questions.

```
def rec(x) :
    if len(x) == x[0] :
        return x
    return rec(x[2:] + [x[0]])
```

[2,1,5]+[3]
[5,3]+[2]
[2]+[5]

23. What will the function call `rec([3, 3, 2, 1, 5])` return?

a) `[5, 3, 3, 2, 1]`

b) `[5, 3, 1]`

c) `[2, 5]`

d) `RecursionError will be raised due to maximum recursion depth being exceeded.`

24. What will the function call `rec([4, 1, 3, 2, 7])` return?

a) `[7, 4, 1, 3, 2]`

b) `[7, 1, 2]`

c) `[3, 7]`

[3,2,7][4]
[7,4][3]
[3][7]

e.g. a=[1,2], a[2:]=[]

d) `RecursionError will be raised due to maximum recursion depth being exceeded.`

25. The following is a recursive function to calculate the sum of a list of numbers.

Example usage:

```
sum([]) ➔ 0
sum([1]) ➔ 1
sum([1, 2]) ➔ 3                    # 1 + 2
sum([1, 2, 3]) ➔ 6                 # 1 + 2 + 3
```

```
def sum(nums) :
    total = 0
    if len(nums) == 0  :
        return 0
    elif len(nums) == 1:
        return nums[0]
    return ## TODO: what goes here
```

Which code fragment will correctly complete the function above?

a)
```
(sum(nums[:len(nums) // 2]) +
 sum(nums[len(nums) // 2:]))
```

b)
```
(sum(nums[1:len(nums) / 2]) +
 sum(nums[len(nums) / 2:-1]))
```

c) `sum(nums[1:len(nums)]) + sum(nums[len(nums):-1])`

d) `sum(nums[1:]) + sum(nums[:-1])`

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

```
class BankAccount(object) :
    def __init__(self, account_number,
                 initial_deposit, over_draft) :
        """
        Parameters:
            account_number (str): Bank account number
                                  for this account.
            initial_deposit (int): Initial amount deposited
                                   into this account.
            over_draft (int): Over draft limit for this
                              account.
        """
        self._account_number = account_number
        self._balance = initial_deposit
        self._over_draft = over_draft

    def credit(self, amount) :
        """Deposit 'amount' into this account."""
        ## code block 1 ##

    def debit(self, amount) :
        """Withdraw 'amount' from this account.
           Return True if account balance and over draft
                  limit will allow withdrawal;
                  False otherwise.
        """
        ## code block 2 ##
```

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

a) `_balance += amount`

b) `_balance() += amount`

c) `self._balance += amount`

d) `self._balance() += amount`

e) None of the statements above are correct.

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

a)
```
if self._balance + self._over_draft > amount :
    self._balance -= amount
    return True
else :
    return False
```

b)
```
if _balance + _over_draft > amount :
    _balance -= amount
    return True
else :
    return False
```

c)
```
return self._balance -= amount
```

d)
```
return ._balance -= amount
```

e) None of the code blocks above are correct.

28. After the following statement is executed:

```
account = BankAccount("123", 100, 100)
```

Which of the following statements will cause the `debit` method to return `False`? (Assume that **`## code block 1 ##`** and **`## code block 2 ##`** contain the correct code.)

a)
```
account.debit() - 250
```

b)
```
account.debit(250)
```

c)
```
debit(account) - 250
```

d)
```
debit(account, 250)
```

e) None of the statements above will cause the debit method to return False.

The next three questions refer to the following function definition, which is missing three lines of code. The function reads raw athlete score data from a file and calculates the athlete's result. The following is an example of a data file (scores.txt).

```
1.1 1.1 1.1 1.1
2 2 2 2
3 3
4.0 4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8
```

Each line of the file represents the scores that an athlete achieved in each round of a competition. These scores may be integer or floating point values and are separated by one space. Athletes may have different numbers of scores. The athlete's result for the competition is the average of all of their scores. 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(scores, results) :
    with open(scores, "r") as scores_data, \
         open(results, "w") as results_data :
        for athlete in scores_data :
            athlete = athlete.strip()
            ## line 1 ##
            total = 0
            ## line 2 ##
            ## line 3 ##
            results_data.write(str(result) + '\n')
```

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

```
process('scores.txt', 'results.txt')
```

Would result in the following data being saved to results.txt.

```
1.1
2.0
3.0
4.4
```

When answering questions 30 and 31, assume that the correct code has been implemented from the previous question(s).

29. What is the required code for **`## line 1 ##`**?

a) `athlete.split(" ")`

b) `athlete.split(",")`

c) `athlete_scores = athlete.split(" ")`

d) `athlete_scores = athlete.split(",")`

30. What is the required code for **`## line 2 ##`**?

a)
```
for score in athlete :
    total += score
```

b)
```
for score in athlete :
    total += float(score)
```

c)
```
for score in athlete_scores :
    total += score
```

d)
```
for score in athlete_scores :
    total += float(score)
```

31. What is the required code for **`## line 3 ##`**?

a) `result = total / len(athlete)`

b) `result = total / len(athlete_scores)`

c) `result = result / len(athlete)`

d) `result = result / len(athlete_scores)`

The next five questions refer to the following class definitions.

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

    def m1(self, x) :
        return self.m2(x) * 2

    def m2(self, x) :
        return x + 1


class B(A) :
    def m2(self, y) :
        return self._x + y


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

    def m1(self, x) :
        return self._x - self._y


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

    def m1(self, y)  :
        return self._y - y

    def m2(self, x) :
        return super().m2(x) + x

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

b.m1(3)
b=B(2)
b._x=2
m1(3)

c.m2(3)
c=C(2,4)
x=2, y=4
5

d.m1(2)
d=D(1,3)
x=1, y=3
x=4,y=3

d.m2(2)
d=D(1,3)
x=1
x=4, y=3
m2(2)
4+2+2

a=A(3)
self.m2(2)*2=6
self.m2(2)=2+1=3

32. What does `a.m1(2)` return?

a) 2

b) 4

c) 5

d) 6

33. What does `b.m1(3)` return?

a) 4

b) 5

c) 6

d) 10

34. What does `c.m2(3)` return?

a) 3

b) 4

c) 5

d) 6

35. What does `d.m1(2)` return?

a) 0

b) 1

c) 5

d) 6

36. What does `d.m2(2)` return?

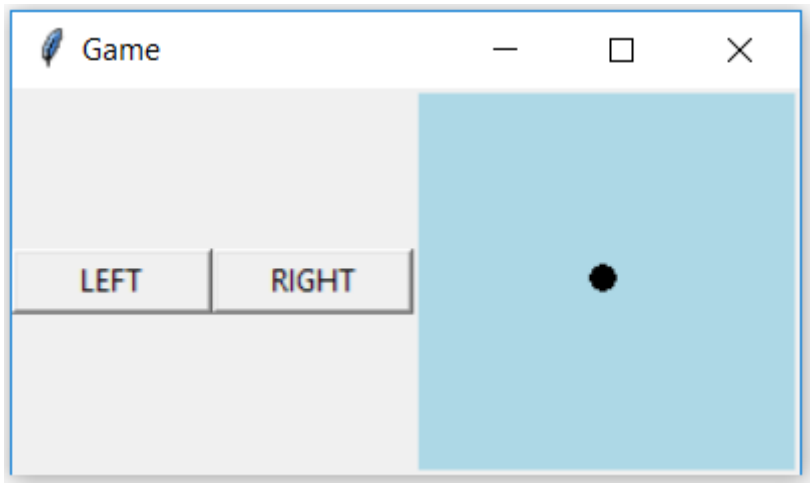
a) 5

b) 6

c) 7

d) 8

The next two questions relate to the following partial implementation of a GUI. The application has two buttons in a frame and custom canvas on which the circle can be moved left or right by clicking on the buttons. The completed GUI is shown in the image below. The code is provided on the next page.



```
import tkinter as tk


class Screen(tk.Canvas):
    def __init__(self, parent):
        super().__init__(parent, bg="light blue",
                         width=150, height=150)
        self._x, self._y = (150 / 2, 150 / 2)
        self._redraw()

    def _redraw(self):
        """Redraw the screen after a move."""
        self.delete(tk.ALL)
        ## code block 1 ##

    def _move(self, dx, dy):
        """Move the circle by a given amount."""
        self._x += dx
        self._y += dy
        self._redraw()

    def move_left(self):
        self._move(-5, 0)

    def move_right(self):
        self._move(5, 0)


class Controls(tk.Frame):
    BUTTON_WIDTH = 10

    def __init__(self, parent, left, right):
        """Parameters:
            parent (Tk): Window for widget.
            left (method): Callback for "left button".
            right (method): Callback for "right button".
        """
        super().__init__(parent)
        ## code block 2 ##


class GameApp(object):
    def __init__(self, master):
        master.title("Game")
        screen = Screen(master)
        controls = Controls(master, screen.move_left,
                            screen.move_right)
        controls.pack(side=tk.LEFT)
        screen.pack(side=tk.LEFT, expand=True,
                    fill=tk.BOTH)
```

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

a)
```
coords = (self._x - 5, self._y - 5,
          self._x + 5, self._y + 5)
create_oval(coords, fill="black", width=0)
```

b)
```
coords = (self._x - 5, self._y - 5,
          self._x + 5, self._y + 5)
self.create_oval(coords, fill="black", width=0)
```

c)
```
coords = (self._x - 5, self._y - 5,
          self._x + 5, self._y + 5)
Canvas.create_oval(coords, fill="black", width=0)
```

d)
```
coords = (self._x - 5, self._y - 5,
          self._x + 5, self._y + 5)
super().create_oval(coords, fill="black", width=0)
```

e) More than one of the above is correct.

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

a)
```
leftBtn = tk.Button(self, text="LEFT",
                    width=10, command=left())
leftBtn.pack(side=tk.LEFT)
rightBtn = tk.Button(self, text="RIGHT",
                     width=10, command=right())
rightBtn.pack(side=tk.LEFT)
```

b)
```
leftBtn = tk.Button(self, text="LEFT",
                    width=10, command=Screen.left())
leftBtn.pack(side=tk.LEFT)
rightBtn = tk.Button(self, text="RIGHT",
                     width=10, command=Screen.right())
rightBtn.pack(side=tk.LEFT)
```

c)
```
leftBtn = tk.Button(self, text="LEFT",
                    width=10, command=self.left())
leftBtn.pack(side=tk.LEFT)
rightBtn = tk.Button(self, text="RIGHT",
                     width=10, command=self.right())
rightBtn.pack(side=tk.LEFT)
```

d)
```
leftBtn = tk.Button(self, text="LEFT",
                    width=10, command=left)
leftBtn.pack(side=tk.LEFT)
rightBtn = tk.Button(self, text="RIGHT",
                     width=10, command=right)
rightBtn.pack(side=tk.LEFT)
```

e) More than one of the above is correct.

39. What is the time complexity, in terms of the length of the list of values, of the following function that returns double the total of all the values 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 double(values) :
    """Double the sum of all numbers in 'values'."""
    result = 0
    for element in values :
        result += element
    for element in values :
        result += element
    return result
```

a) Constant

b) Logarithmic

c) Linear

d) Quadratic

e) Exponential

40. What is the time complexity, in terms of the value of exponent, of the following function that calculates exponentiation? You may assume logical comparisons and arithmetic operations are all constant time operations.

```
def exponentiation(num, exponent) :
    """Calculate 'num' raised to the 'exponent'."""
    result = 1
    while exponent > 0 :
        if exponent % 2 == 0 :
            num = num * num
            exponent /= 2
        else :
            exponent -= 1
            result *= num
    return result
```

a) Constant

b) Logarithmic

c) Linear

d) Quadratic

e) Exponential

END OF EXAMINATION