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, 2017

## CSSE1001/7030 Introduction to Software Engineering I

*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

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.**

Answer all questions on the supplied Multiple Choice / True False answer sheet.

**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 `(6.0 + 11) / 2` evaluate to?

   a) 8

   b) 8.0

   [✓ c) 8.5]

   d) Error

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

   a) 1

   b) 4

   c) 4.5

   [✓ d) 5]

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

   [✓ a) (1, 2, 3, 4)]

   b) (4, 6)

   c) (4, 5, 5, 6)

   d) Error

4. What is the result of `'a' < 'b'`?

   a) 'a'

   [✓ b) True] cuz a come before b, therefore True

   c) False

   d) Error

5. What is the value of x after the following is evaluated?

```
x = [3,2,1]
y = x
y[1] = 5
```

   a) [ ]

   [✓ b) [3, 5, 1]] mutable variable will change
   if change the element in the variable it assigned to

   c) [5, 2, 1]

   [✓ d) [3, 2, 1]]

6. What is the value of z after the following is evaluated?

```
y = ['a', 'b']
z = ['t']
y.extend(['c'])
z.append(y)
```

a) ['t', ['a', 'b', 'c']]

b) ['t', 'a', 'b', 'c']

c) ['t', ['c', 'a', 'b']]

d) ['t', 'c', 'a', 'b']

7. What will be the value of x after evaluating these statements?

```
x = [1, 2, 3, 4]
x.append(x.pop(1))
x.insert(2, x.pop(3))
```

1,3,4,2

1,3,2,4

a) [1, 2, 3, 4]

b) [2, 4, 3, 1]

c) [1, 3, 2, 4]

d) [3, 2, 1, 4]

8. What is the value of d after the following is evaluated?

```
d = {'a':1, 'b':2}
d[['a','b']] = 34
```

Note: for a dictionary the ordering of the elements does not matter.

a) {'a':1, 'b':2, ['a','b']:34}

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

c) {'a':3, 'b':4}

d) Error

9. What is the value of y after the following is evaluated?

```
d = {'Brisbane': {2013:24.1, 2014:24.2},
     'Adelaide': {2012:22.1, 2013:22.6, 2014:22.8}}
y = d.get('Brisbane', {}).get(2012)
```

a) None

b) {}

c) [24.1, 22.8]

d) {2013:24.1, 2014:24.2}

e) Error

10. What is the value of y after the following is evaluated?

```
d = {'Brisbane': {2013:24.1, 2014:24.2},
     'Adelaide': {2012:22.1, 2013:22.6, 2014:22.8}}
y = d.get('Adelaide', {}).get(2012)
```

a) None

b) {}

c) 22.1

d) {2012:22.1, 2013:22.6, 2014:22.8}

e) Error

11. The following function outputs the power given the voltage (as a float) and the current (as a float).

```
def power(voltage, current) :
    print(str(voltage * current) + " W")
```

What is the return type of this function?

a) str

[✓ b) None] as it only print the value, it return None

c) float

d) int

12. If you wished to validate input to guarantee that a user only entered one of the following values: 'a', 'b', 'c' or 'd', based on the following input statement:

```
value = input("Enter one of: 'a', 'b', 'c' or 'd' ")
```

Which of the following if statements would correctly test that the input was valid?

a) if value == ("a" or "b" or "c" or "d") : only return 'a' can return True

b) if value in "abcd" :

c) if value not in "abcd" :

d) if value in ("a", "b", "c", "d") :

13. For the following function:

```
def logic(x, y, z) :
    return x and y or z
```

What is returned by `logic(False, False, True)`

a) True

b) False

c) 1

d) 0

14. What will be returned at the end of this function, assuming count is an int?

```
def test(count) :
    x = True
    while count > 0   :
        if count > 100 :
            x = True
        else :
            x = False
        count -= 1
    return x
```

a) True if count is less than or equal to zero, otherwise False

b) False if count is less than or equal to zero, otherwise True

c) True if count is in the range of 1 to 100, otherwise False

d) False if count is in the range of 1 to 100, otherwise True

15. In general, why are docstrings necessary when developing good quality code?

a) The person who reads the code may not be familiar with the programming language so will need extra guidance to help them understand what it does.

b) To inform the users of a class, method or function about what it does, how to use it, and what its input and output types are.

c) The docstring provides a description of the algorithm implemented by the class, method or function code.

d) Python uses the docstring to check that parameters passed to a function or method are of the correct type.

16. Why should inline comments be used?

a) The person who reads the code may not know how to program so the comments are necessary to describe in detail what the code does.

b) To allow the programmer to add their own personal style to the structure of the code.

c) They should not be used as the code should be self-explanatory.

d) To explain complex or tricky parts of the code.

17. Why is the use of global variables considered to be bad practice?

a) Global variables use up more memory than local variables.

b) Accessing global variables is inefficient and slower than accessing local variables.

c) Global variables can be modified by any part of the program, making it difficult to trace errors.

d) They are not bad practice as they are a useful way to share data between different parts of a program.

18. For the following block of code:

```
words = ["pick", "an", "answer"]
print(words[0])
print(len(words[0]))
print(words[1])
print(len(words[1]))
print(words[2])
print(len(words[2]))
```

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

a) an if statement

b) a function

c) a class

d) a loop

19. For the following block of code:

```
names = ["John", "Marsha", "Bob", "Petunia", "Willy"]
scores = [90, 80, 100, 99, 65]
genders = ["male", "female", "male", "female", "other"]
heights = ["tall", "medium", "medium", "tall", "short"]
preferences = [("male", "tall"), ("male", "tall"),
               ("female", "short"), ("female", "medium"),
               ("other", "short")]
```

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

a) an if statement

b) a function

c) a class names, scores, genders, heights, preferences can be the variables of the class

d) a loop

20. For the following function:

```
def rec(x):
    if x==1:
        return x
    else:
        return rec(x-1)*x
```

rec(3) * 4
rec(2)*3*4
rec(1)*2*3*4

What will `rec(4)` return?

a) 0

b) 4

c) 8

d) 24

21. The following is a recursive function to calculate the product of a list of numbers.

Example usage:

```
product([]) = 0
product([4, 2]) = 8                   # 4 * 2
product([3, 2, 4]) = ~~18~~ 24            # 3 * 2 * 4
product([2, 3, 4, 5, 6]) = 720     # 2 * 3 * 4 * 5 * 6
```

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

Which line of code will correctly complete the function above?

```
a) (product(nums[:len(nums) // 2]) *
    product(nums[len(nums) // 2:]))
```

```
b) product(nums[1:]) * product(nums[:-1])
```

```
c) (product(nums[1:len(nums) / 2]) *
    product(nums[len(nums) / 2:-1]))
```

```
d) (product(nums[1:len(nums)]) *
    product(nums[len(nums):-1]))
```

The next two questions refer to the following function definition.

```
def get_days(years) :
    total_days = 0

    while years >= 0 :
        total_days += 365
        years -= 1

    return total_days
```

22. When the following code is executed, what, if any, error will be thrown?

```
years = input("How many years to convert to days? ")
days = get_days(years)
print("You entered ", years, "years.")
print("That is {} days.".format(days))
```

a) ValueError

b) NameError

[✓ c) TypeError] the input is string, it has to change to integer

d) No error will be thrown.

23. What will the following function call return? (If you determined that an error would be thrown in the previous question, assume that it has been fixed.)

```
get_days(2)
```

a) 0

b) 365

c) 730

d) 1095

365 1
+365 0
+365 -1

The next three questions (on the following page) refer to the following function definition, which is missing three lines of code. This function reads data from a file and calculates averages. The following is an example of a data file (values.txt).

```
name1 :
1.2
2.3

name2 :

2.4
name3:
1.7

1.9
end:
```

The file is divided into sections with each section starting with a (non-empty) name followed by zero or more spaces followed by a colon. The name of a section is followed by one or more lines containing floating point numbers. Blank lines may appear anywhere in the file. The last line of the file is `end:`.

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

```
def get_averages(filename) :
    file = open(filename, 'r')
    averages = {}
    name = None
    for line in file :
        line = line.strip()
        if line == '' :
            ## line 1 ##
        if line.endswith(':') :
            if name is not None :
                ## line 2 ##
            name = line[:-1].strip()
            num = 0
            total = 0.0
        else :
            ## line 3 ##
    file.close()
    return averages
```

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

```
print(get_averages('values.txt'))
```

Would result in the following being output.

```
{'name1': 1.75, 'name2': 2.4, 'name3': 1.8}
```

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

    a) break

    [✓ b) continue]

    c) name = None

    d) averages[name] = 0

    e) More than one of the above is correct.

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

    a) name = None

    b) averages[name] = 0

    c) averages[name] = total / num

    d) averages[name] = total / (num + 1)

    e) More than one of the above is correct.

26. What is the required code for `## line 3 ##`? Recall that a semi-colon allows you to write two or more statements on one line.

    a) num += 1; total += line

    b) num += 1; total += float(line)

    c) averages[name] = 0; num = 0; total = 0

    d) averages[name] = total / num; num += 1

    e) More than one of the above is correct.

The next five 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 2*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 self.x + self.y

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

    def g(self, y)  :
        return self.y + y

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

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

a.f(2)
a = A(3), self.x = 3
f(2)
self.g(2)-1
2*2-1 = 3

c.g(3)
c = C(2,4)
self.x (inherite A) = 2
self.y = 4
g(3) --> class B
g(3) = self.x+y = 2+3 = 5

b.f(3)
b = B(2)
self.x = 2
f(3)
self.g(3) - 1
self.x+y-1 = 2+3-1

d.f(2)
d = D(1,3)
self.x = 1
self.x = 1+3 = 4
self.y = 3
f(2)
B.f(2) - 2
self.g(2) - 1 - 2
self.y + 2 - 1 - 2 = 2

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

a) 2

b) 3

c) 4

d) 5

a = A(3)  self.x = 3
g(2)
2*2=4

28. What does `a.f(2)` return?

    a) 2

    [✓ b) 3]

    c) 4

    d) 5

29. What does `c.g(3)` return?

    a) 2

    b) 3

    c) 4

    [✓ d) 5]

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

    a) 2

    [✓ b) 3]

    [✓ c) 4]

    d) 5

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

    [✓ a) 2]

    b) 3

    c) 4

    d) 5

The next three questions refer to the following partial definition of a GoCard class.

```
class GoCard :
    def __init__(self, user, initial_balance) :
        """
        Parameters:
            user (str): Name of the GoCard user.
            initial_balance (float): Initial amount
                                     loaded on to GoCard.
        """
        self._user = user
        # the balance on the card in dollars
        self._balance = initial_balance

    def update_balance(self, value) :
        """Update the balance with 'value'.
            value > 0 - the card is topped up
            value < 0 - the value of the trip
        """
        ## line 1 ##

    def get_balance(self) :
        """(float) Return the balance."""
        ## line 2 ##
```

Assume that the following has been evaluated.

```
Fred = GoCard('Fred', 100)
```

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

a) `balance += value`

b) `_balance += value`

c) `self.balance += value`

[✓ d) `self._balance += value`]

e) More than one of the above is correct.

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

a) `print self.balance`

b) `return self.balance`

c) `print self._balance`

[✓ d) `return self._balance`]

e) More than one of the above is correct.

34. Which of the following correctly updates the balance by -$2.75 for the object fred?

    a) `update_balance(fred) - 2.75`

    b) `update_balance(fred, -2.75)`

    c) `fred.update_balance(-2.75)`

    d) `fred.update_balance() - 2.75`

    e) `More than one of the above is correct.`

35. What is the advantage of using inheritance and polymorphism in designing a class hierarchy?

    a) They provide a mechanism to extend a program by adding new functionality by inheriting from a class and overriding one or more of the methods defined in the super class.

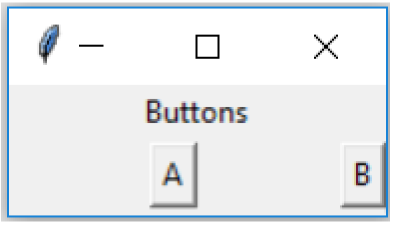
    b) They provide a mechanism to reuse code written in a super class, reducing the amount of code and testing that needs to be done for an application.

    c) They provide a mechanism to extend standard libraries so that programmers working on other projects can take advantage of the new features added to the libraries.

    d) They provide a mechanism to simplify class design by enabling the code to be split into multiple files.

    e) None of the answers above are valid descriptions of an advantage of using inheritance and polymorphism.

The next two question relate to the following partial definitions. In a GUI application we decide we need a widget that contains two buttons and that this widget is to appear within the main window of the application below the label as shown in the image below.



```
class ButtonsFrame(Frame) :
    def __init__(self, parent) :
        Frame.__init__(self, parent.root)
        b1 = Button(self, text= "A")
        b2 = Button(self, text = "B")
        ## lines 1 and 2 ##

class MainWindow(object) :
    def __init__(self, root) :
        self.root = root
        Label(root, text="Buttons").pack()
        bf = ButtonsFrame(self)
        ## line 3 ##
```

36. What is the required code for `## lines 1 and 2 ##`?

```
a) b1.pack(side=LEFT, expand=1)
   b2.pack(side=LEFT)

b) b1.pack(side=LEFT, expand=1)
   b2.pack(side=LEFT, expand=1)

c) b1.pack(side=LEFT, fill=BOTH)
   b2.pack(side=LEFT, fill=BOTH)

d) b1.pack(side=LEFT, fill=BOTH)
   b2.pack(side=LEFT, fill=X)

e) More than one of the above is correct.
```

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

```
a) bf.pack(expand=1)

b) bf.pack(fill=BOTH, expand=1)

c) bf.pack()

d) bf.pack(fill=BOTH)

e) More than one of the above is correct.
```

The next two question relate to the following function definition. The function tests to see if the list has repeated elements.

```
def has_repeats(list) :
    """Return True iff 'list' has repeated elements."""
    size = len(list)
    for i in range(size-1) :
        element = list[i]
        for j in range(i+1, size) :
            if element == list[j] :
                return True
    return False
```

38. What is the time complexity, in terms of the length of the list for the function above? You may assume accessing elements of a list, and the arithmetic operations and tests are all constant time operations.

    a) Constant

    b) Logarithmic

    c) Linear

    d) Quadratic

    e) Exponential

39. How many tests would be required to adequately demonstrate that the `has_repeats` function performs as expected?

    a) Two: a list with no repeated elements and a list with repeated elements, e.g. [1, 2, 3, 4] and [1, 2, 2, 3].

    b) Three: a list with no elements, a list with one element, and a list with multiple elements would adequately test the function, e.g. [ ], [1, 2, 3, 4] and [1, 2, 2, 3].

    c) Four: a list with no elements, a list with one element, a list with multiple non-repeated elements, and a list with repeated elements would adequately test the function, e.g. [ ], [1], [1, 2, 3, 4] and [1, 2, 2, 3].

    d) Five: a list with no elements, a list with one element, a list with multiple non-repeated elements, a list with adjacent repeated elements, and a list with non-adjacent repeated elements would adequately test the function, e.g. [ ], [1], [1, 2, 3, 4], [1, 2, 2, 3] and [1, 2, 3, 2].

    e) Six: a list with no elements, a list with one element, a list with multiple non-repeated elements, a list with adjacent repeated elements, a list with non-adjacent repeated elements, and a list with multiple sets of repeated elements would adequately test the function, e.g. [ ], [1], [1, 2, 3, 4], [1, 2, 2, 3], [1, 2, 3, 2] and [1, 2, 2, 3, 4, 3].

40. What is the time complexity, in terms of the length of the string, of the following function that tests a string to see if it is a palindrome (i.e. a string that can be read the same way in either direction) such as "radar"? You may assume accessing elements of a string, calculating the length of a string and the arithmetic are constant time operations.

```
def is_palindrome(string) :
    """Return True iff 'string' is a palindrome."""
    size = len(string)
    half = size / 2
    i = 0
    while i < half:
        if string[i] != string[size-1-i] :
            return False
        i += 1
    return True
```

a) Constant

b) Logarithmic

c) Linear

d) Quadratic

e) Exponential

**END OF EXAMINATION**