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, 2021

CSSE1001/7030 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 Closed Book examination - no written materials permitted
No calculators 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)

None

Materials To Be Supplied To Students:

1 x Multiple Choice Answer Sheet

Instructions To Students:

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

Answer all questions on the provided multiple choice 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) `11.5`

   b) `11.0`

   c) `11`

   d) `8.5`

2. What does the expression `0 > -1 < -2` evaluate to?

   a) `True`

   b) `False`

   c) `Error`

3. What does the expression `0 < -1 or -4 < -2` evaluate to?

   a) `True`

   b) `False`

   c) `Error`

4. What does the expression `0 < 10 and not (4 < 2)` evaluate to?

   a) `True`

   b) `False`

   c) `Error`

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

   a) `(1, 2)`

   b) `(3, 2)`

   c) `(1, 2, 2)`

   d) `[(1, 2), (2, )]`

   e) `Error`

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

   a) `[1, 2, 3]`

   b) `[2, 4, 3]`

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

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

   e) `Error`

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

```
s1 = "Monty"
s2 = "Python"
s3 = "Spam"
s3 = s1
s3 = "Parrot"
s4 = s1 + s2
```

a) `"ParrotPython"`

b) `"MontyPython"`

c) `"SpamPython"`

d) `Error`

8. After the assignment `s1 = "Monty " + "Python" + " Sketch"`, which of the following statements assigns `"Python"` to `s2`?

a) `s2 = s1[1]`

b) `s2 = s1[6:12]`

c) `s2 = s1[7:12]`

d) `s2 = s1[-13:-8]`

e) More than one of the above is correct.

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

```
x = ['a', 0, 'b']
y = x
y[2] = 0
```

a) `['a', 0, 0]`

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

c) `0`

d) `Error`

10. What is the consequence of executing the following statements?

```
x = (1, 2)
x += (3, 4)
```

a) Tuples are immutable, the `+=` operator will cause an error.

b) The second statement will add the values 3 and 4 to the tuple stored in `x`.

c) Tuples are immutable, the second statement will assign the tuple `(3, 4)` to `x`.

d) The second statement will create a new tuple that contains the contents of `x` and the tuple to the right of the `+=` operator. This new tuple is assigned to `x`.

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

```
d = {'CA':'Maple Syrup', 'AU':'Vegemite',
     'NZ':'Pavlova', 'FR':'Escargot'}
d['SP'] = 'Churro'
d.get('JP', 'Sushi')
```

a) `{'JP': 'Sushi'}`

b) `{'SP': 'Churro'}`

c) `{'CA': 'Maple Syrup', 'AU': 'Vegemite', 'NZ': 'Pavlova', 'FR': 'Escargot'}`

d) `{'CA': 'Maple Syrup', 'AU': 'Vegemite', 'NZ': 'Pavlova', 'FR': 'Escargot', 'SP': 'Churro'}`

e) `{'CA': 'Maple Syrup', 'AU': 'Vegemite', 'NZ': 'Pavlova', 'FR': 'Escargot', 'SP': 'Churro', 'JP': 'Sushi'}`

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

```
x = 0
if x == 0:
    print("zero")
elif x >= 0:
    print("positive")
elif x <= 0:
    print("negative")
```

a) `zero`

b) `positive`

c) `negative`

d)
```
zero
positive
negative
```

e) `Error`

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

```
x = 0
y = 1
while x < 5:
    x += 1
    y *= x
```

a) 1

b) 24

c) 120

d) 720

e) The logic is an infinite loop.

14. What is output after calling the function `f1`?

```
def f1() -> None:
    v = 8
    x = f2(v)
    print(v)

def f2(x: int) -> int:
    v = 2
    return x // v

f1()
```

a) 8

b) 4

c) 2

d) 0

e) Error

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

```
def f(x: list[int]) -> int:
    y = 0
    z = 0
    for i in x:
        y += i
        z += 1
    return y / z

a = [10, 20, 30, 40]
print(f(a))
```

a) 10.0

b) 25.0

c) 33.33333333336

d) 40.0

e) Error

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

```
def f(x: list[int]) -> None:
    for i in range(len(x) // 2):
        y = x[i]
        x[i] = x[len(x) - i - 1]
        x[len(x) - i - 1] = y

a = [1, 2, 3, 4, 5]
f(a)
print(a)
```

a) `[1, 2, 3, 4, 5]`

b) `[1, 4, 3, 2, 5]`

c) `[5, 2, 3, 4, 1]`

d) `[5, 4, 3, 2, 1]`

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: list[int]) -> tuple[int, int]:
    x = lst[0]
    y = lst[0]
    for i in lst:
        if x > i:
            x = i
        elif y < i:
            y = i
    return x, y
```

a) It returns the sum of the half of the list made up of the smallest numbers, and the sum of the half of the list made up of the largest numbers.

b) It raises an error because the type hint indicates it returns a tuple but the code returns two values.

c) It returns the smallest and largest values found in the list.

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

18. Which of the following statements is correct regarding global variables?

a) Global variables save memory, as they are stored once for the entire program, and should be used in good software design.

b) Global variables are shared across the entire program, meaning it is more difficult to determine when their value has changed.

c) Global variables are inefficient because every function call has to create a copy of the variables in case they are used in the function.

d) Global variables provide a way of describing all important variables in a program in a single location, so assist with writing good documentation.

e) Both (b) and (c).

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

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

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

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

    d) To identify potential errors that may be encountered in a block of code.

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

```
def f(x: int) -> None:
    if x <= 0:
        raise ValueError("value must be positive")
    while x > 0:
        print(x, end=',')
        x -= 1

try:
    f(-5)
except ValueError as e:
    print(str(e))
```

    a) `value must be positive`

    b) `-5,-4,-3,-2,-1,0,`

    c) `5,4,3,2,1,`

    d) There is no output because the while loop is never entered in function `f`.

    e) The while loop in function `f` is an infinite loop, so the output never stops.

21. For the following function:

```
def r(x: int, y: int) -> int:
    if x == 0:
        return x * y
    return r(x-1, y) + y
```

What will `r(4, 2)` return?

    a) `10`

    b) `8`

    c) `4`

    d) `0`

    e) `RecursionError: maximum recursion depth exceeded`

22. For the following block of code:

```
TITLE = 0
RELEASE_YEAR = 1
NOMINATIONS = 2
AWARDS = 3

movies = [("Parasite", 2019, 6, 4),
          ("Little Women", 2019, 6, 1),
          ("1917", 2019, 10, 3),
          ("Bohemian Rhapsody", 2018, 5, 3),
          ("Black Panther", 2018, 7, 3)]

def num_awards(movie: tuple[str, int, int, int]) -> int:
    return movie[AWARDS]

def num_nominations(movie: tuple[str, int, int, int]) -> int:
    return movie[NOMINATIONS]

def add_award(movie: tuple[str, int, int, int], new_awards: int) \
        -> tuple[str, int, int, int]:
    return (movie[TITLE], movie[RELEASE_YEAR],
            movie[NOMINATIONS], movie[AWARDS] + new_awards)

movies[2] = add_award(movies[2], 2)
```

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

a) dictionary

b) while loop

c) if statement

d) function

e) class

23. For the following block of code, assume that `Position`, `Entity`, `Player` and `Zombie` are the classes that were defined in assignment two.

```
class Locations:
    def __init__(self):
        self._positions = []
        self._entities = []

    def add_entity(self, position: Position, entity: Entity) -> None:
        self._positions.append(position)
        self._entities.append(entity)

    def get_entity(self, position: Position) -> Optional[Entity]:
        for i, p in enumerate(self._positions):
            if p == position:
                return self._entities[i]

l = Locations()
l.add_entity(Position(1, 1), Player())
l.add_entity(Position(2, 2), Zombie())
p = l.get_entity(Position(1, 1))
```

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

a) dictionary

b) tuple

c) while loop

d) if statement

e) function

24. The following is an extract of code from the `Inventory` class from assignment two. Part of the implementation of the `step` method has been removed.

```
class Inventory:
    def __init__(self):
        self._items = []

    def step(self):
        """
        When this method is called, the lifetime of every item
        stored within the inventory should decrease. Any items
        in the inventory that have exceeded their lifetime should
        be removed.
        """
        new_items = ## Fragment 1 ##
        for item in self._items:
            item.hold()
            if item.get_lifetime() > 0:
                new_items.append(item)
        ## Fragment 2 ##
```

Which code fragments below will correctly complete the function above?

a) Fragment 1 is: `self._items`
   Fragment 2 is: `self._items = new_items`

b) Fragment 1 is: `self._items`
   Fragment 2 is: `new_items = self._items`

c) Fragment 1 is: `[]`
   Fragment 2 is: `self._items = new_items`

d) Fragment 1 is: `[]`
   Fragment 2 is: `new_items = self._items`

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

The next two questions refer to the following function definition, which is missing two fragments of code. It is a recursive function to find all the positive whole number divisors of a number between the parameters `i` and `number`. The list of divisors will be returned via the parameter `result`.

```
def positive_divisors(number: int, i: int, result: list[int]) -> None:
    """
    Precondition: number > 0 and i > 0
    """
    if number % i == 0:
        ## Fragment 1 ##
    if number > i:
        ## Fragment 2 ##
```

Example usage:

```
r = []
positive_divisors(2, 1, r)
r == [1, 2]
r = []
positive_divisors(9, 1, r)
r == [1, 3, 9]
```

When answering question 26, assume that the correct code has been implemented from the previous question.

25. What code is required at **`## Fragment 1 ##`**?

a) `return 0`

b) `return i`

c) `result += i`

d) `result.append(i)`

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

26. What code is required at **`## Fragment 2 ##`**?

a) `positive_divisors(number, i-1, result)`

b) `positive_divisors(number, i+1, result)`

c) `positive_divisors(number, i, result)`

d) `positive_divisors(number, 1, result)`

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

The next two questions refer to the following function definition, which is missing two fragments of code. The function reads student assessment results for a course from a file. It then calculates the total mark achieved by each student in the course and saves this to another file. The following is an example of a data file (marks.csv).

```
41234567,10,9,14,18,42
40000000,5,5,7,8,12
41111111,8,6,12,14,21
```

The first value on each line of the file is a student number. All the following data items on the line are the marks the student achieved in each assessment in the course. These values may be integer or floating point and are separated by a comma. You may assume that the data in the input file is correctly formatted. The results are written to an output file where each line has a student number and total mark for the course.

The definition of the `process` function, with two missing code fragments, is given below.

```
def process(marks: str, results: str) -> None:
    with open(marks, 'r') as mark_data, \
         open(results, 'w') as result_data:
        for entry in mark_data:
            student = entry.split(',')
            ## Fragment 1: Initialise student's score ##
            ## Fragment 2: Calculate student's total score ##
            result_data.write(student[0] + ','
                              + str(student_score) + '\n')
```

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

```
process('marks.csv', 'results.csv')
```

would result in the following data being saved to `results.csv`.

```
41234567,93.0
40000000,37.0
41111111,61.0
```

When answering question 28, assume that the correct code has been implemented from the previous question.

27. What code is required at **`## Fragment 1: Initialise student's score ##`**?

a) `student_score = float(student[0])`

b) `student_score = student[0]`

c) `student_score = 1`

d) `student_score = 0`

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

28. What code is required at **`## Fragment 2: Calculate student's total score ##`**?

a)
```
for i in range(1, len(student)):
    student_score += float(student[i])
```

b)
```
for i in range(1, len(student)):
    student_score += student[i]
```

c)
```
for score in student:
    student_score += float(score)
```

d)
```
for score in student:
    student_score += score
```

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

The following partial definition of a shopping basket class, to be used in an online store, is used in the following three questions.

```
class ShoppingBasket:
    def __init__(self):
        self._items = {} # Dictionary of items in this basket.
                         # Item stock id is the key mapped
                         # to the price of the item.

    def add_item(self, item_id: str, price: float) -> None:
        """Add an item to this basket.

        Parameters:
            item_id: The item's stock id.
            price: The item's price.
        """
       ## Fragment 1 ##

    def total(self) -> float:
    """Return the total price of all items in this basket."""
        total_price = 0
        ## Fragment 2 ##
        return total_price
```

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

a) `self._items.get(item_id, price)`

b) `self._items.set(item_id, price)`

c) `self._items[item_id] = price`

d) `self._items[price] = item_id`

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

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

a)
```
for price in self._items.items():
    total_price += price
```

b)
```
for price in self._items.values():
    total_price += price
```

c)
```
for price in self._items():
    total_price += price
```

d)
```
for price in self._items:
    total_price += price
```

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

31. Assuming that the `ShoppingBasket` class is implemented correctly, which of the following sets of statements is the best approach to output `"The total is 30"`?

a)
```
basket = ShoppingBasket()
ShoppingBasket.add_item(basket, "1", 10)
ShoppingBasket.add_item(basket, "2", 20)
print("The total is", ShoppingBasket.total(basket))
```

b)
```
basket = ShoppingBasket()
ShoppingBasket.add_item("1", 10)
ShoppingBasket.add_item("2", 20)
print("The total is", ShoppingBasket.total())
```

c)
```
basket = ShoppingBasket()
basket.add_item("1", 10)
basket.add_item("2", 20)
print("The total is", basket.total())
```

d)
```
basket = ShoppingBasket()
add_item(basket, "1", 10)
add_item(basket, "2", 20)
print("The total is", total(basket))
```

e) None of these statements will result in `"The total is 30"` being output.

32. When designing classes, what does “respecting the abstraction barrier” mean?

a) It means that if the implementation of a class’ data structure or methods changes, but the interface to the methods does not change, other code can continue to use objects of the class without any changes.

b) It means that a subclass can use the attributes of its superclass without worrying about changes in the design of the superclass.

c) It means that methods should only directly access the private attributes of the class and not call other methods of the class. This avoids passing unnecessary parameters to methods.

d) It means that the class is designed so that most methods make use of most attributes of the class. This indicates that the class is a well-designed simple concept.

e) None of the answers above are valid descriptions of “respecting the abstraction barrier”.

The next four questions refer to the following class definitions and object instantiations.

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

    def f(self, a: int) -> int:
        return self.g(a) + self._x

    def g(self, a: int) -> int:
        return a * 2

class B(A):
    def g(self, a: int) -> int:
        return self._x * a

    def h(self, a: int) -> int:
        return super().g(a)

class C(B):
    def __init__(self, a: int, b: int):
        super().__init__(a)
        self._y = b

    def f(self, a: int) -> int:
        return self.h(self._y) + a

a = A(2)
b = B(3)
c = C(4, 6)
```

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

a) `4`
b) `6`
c) `8`
d) `10`
e) `Error`

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

a) `6`
b) `9`
c) `12`
d) `15`
e) `Error`

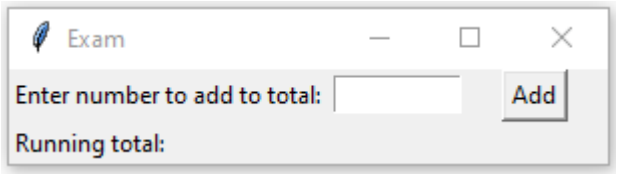
35. What does `b.h(4)` return?

a) `4`
b) `8`
c) `11`
d) `12`
e) `Error`

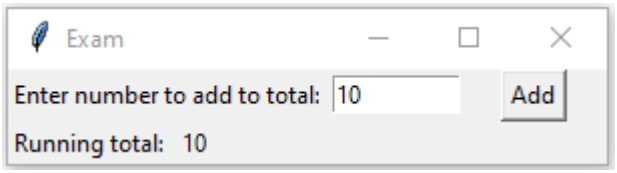
36. What does `c.f(5)` return?

a) `14`
b) `15`
c) `17`
d) `29`
e) `Error`

The next two questions relate to the following simple GUI application. The application has a text field into which users can enter numbers. When started, the GUI appears as in the image below.



When the user presses the Add button, the value in the text field is added to the running total and displayed, as shown in the image below.



The code, with two missing code fragments, is provided below.

```
import tkinter as tk

class Input(tk.Frame):
    def __init__(self, parent: tk.Tk, add_event_handler):
        super().__init__(parent)

        prompt = tk.Label(self, text="Enter number to add to total: ")
        prompt.pack(side=tk.LEFT)

        self._entry = tk.Entry(self, width=10)
        self._entry.pack(side=tk.LEFT)

        ## Fragment 1 ##
        add_button.pack(side=tk.LEFT, padx=20)

    def get_input(self) -> int:
        return int(self._entry.get())


class Add:
    def __init__(self, master: tk.Tk):
        master.title("Exam")
        self._total = 0

        self._input_frame = Input(master, self.add)
        self._input_frame.pack(side=tk.TOP)

        total_label = tk.Label(master, text="Running total: ")
        total_label.pack(side=tk.LEFT)
        self._result = tk.Label(master)
        self._result.pack(side=tk.LEFT)

    def add(self) -> None:
        ## Fragment 2 ##
```

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

a) `add_button = tk.Button(self, text="Add", command=Add())`

b) `add_button = tk.Button(self, text="Add", command=Add.add)`

c) `add_button = tk.Button(self, text="Add", command=self.get_input)`

d) `add_button = tk.Button(self, text="Add", command=add_event_handler)`

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

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

a)
```
self._total += self.get_input()
self._result.config(text=str(self._total))
```

b)
```
self._total += self._input_frame.get_input()
self._result.config(text=str(self._total))
```

c)
```
self._total += self._input_frame._entry.get()
self._result.config(text=str(self._total))
```

d)
```
self._total += self._input_frame.get_input()
self._result.config(text=str(self._input_frame.get_input()))
```

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

The next two questions relate to the following function definition. The function returns a tuple indicating how many positive, zero and negative values were in the list passed as a parameter to `values`.

```
def partition(values: list[int]) -> tuple[int, int, int]:
    positives = 0
    zeros = 0
    negatives = 0

    for value in values:
        if value == 0:
            zeros += 1
        elif value > 0:
            positives += 1
        else:
            negatives += 1

    return (positives, zeros, negatives)
```

39. What is the time complexity, in terms of the logical complexity of the `partition` function? You may assume arithmetic and comparison operations are constant time operations.

    a) O(1) – Constant

    b) O(log n) – Logarithmic

    c) O(n) – Linear

    d) $O(n^2)$ – Quadratic

    e) $O(2^n)$ – Exponential

40. Assume each of the following sets of lists are individually passed as parameters to the `partition` function. Which of these sets of lists, would be the fewest number of inputs that would be required to adequately test all logical paths in the function?

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

    b) [ ], [1], [0], [-1]

    c) [ ], [1], [0], [-1], [1, 0, -1]

    d) [ ], [1], [0], [-1], [1, 0], [0, -1], [1, -1], [1, 0, -1], [1, -2, 0, -1, 2, 0]

    e) [ ], [1], [0], [-1], [1, 0], [0, -1], [1, -1], [1, 0, -1], [1, 0, 2, -1], [1, 0, -1, 0], [1, -2, 0, -1], [1, 0, 2, -1, 0], [1, -2, 0, -1, 2], [1, -2, 0, -1, 0], [1, -2, 0, -1, 2, 0], ["a"], ["a", "b"], ["a", 1]

END OF EXAMINATION