IT5001—*Software Development Fundamentals*

AY21/22 Semester 2 Final Exam

Rules:

- You are allowed **ONE** A4-sized cheat sheet, double-sided, printed or written.
- You are allowed **ONE** blank sheet of A4 paper (in addition to your cheat sheet) for drafts.
- You **cannot** refer to any another document or search for information online.
- You **cannot** access any files on your computer, including `.py` files.
- You **cannot** use any other electronic devices, including smart watches.
- You **cannot** use other tools or IDE, such as IDLE, pycharm, etc. to help you.
- You **cannot** communicate with anyone throughout the exam.

This document contains **12 printed pages**.
You have **2 hours** to complete the exam.
There are **35 questions**.

Section	**Awarded**	**Maximum**
Expression Evaluation		36
Program Tracing		23
Programming		41
Total		100

(Solutions are appended to the end of this document).

Expression Evaluation [2 marks each]

Without using IDLE, determine the results from evaluating the following Python expressions and select the correct option.

1) `6 ** 2 * 0 + 1`
A. `1`
B. `2`
C. `36`
D. `37`
E. Evaluating this expression yields an error

2) `False and True or False and False or True`
A. `True`
B. `False`
C. `None`
D. Evaluating this expression yields an error

3) `'abcdefghi'[2:8:2][2]`
A. `''` (empty string)
B. `'c'`
C. `'e'`
D. `'g'`
E. Evaluating this expression yields an error

4) `{6: 1, 8: 2, 1: 8}[1]`
A. `8`
B. `6`
C. `2`
D. `8: 2`
E. Evaluating this expression yields an error

5) `tuple(map(str, range(0, 3)))`
A. `'(012)'`
B. `'(0, 1, 2)'`
C. `('0123',)`
D. `('012',)`
E. `('0', '1', '2')`

6) `sum([i * 2 for i in range(4, 10, 5)])`
A. `8`
B. `10`
C. `13`
D. `22`
E. `26`

7) `5 * int('4 * 2')`
A. `'40'`
B. `'88888'`
C. `'4*24*2'`
D. `40`
E. Evaluating this expression yields an error

8) `'abcde'[2:5][1][0][0][1:0]`
A. `''` (empty string)
B. `'d'`
C. `'cde'`
D. `[]`
E. Evaluating this expression yields an error

(proceed to the next page...)

9) `round(1.5) + round(2.5)`
A. `3`
B. `4`
C. `5`
D. `5.0`
E. Evaluating this expression yields an error

10) `1 > 1 + 1 or 3 > 7 - 6 or 6 > 7 / 0`
A. `True`
B. `False`
C. `None`
D. `ZeroDivisionError`
E. `TypeError`

11) `11 & 4`
A. `0`
B. `1`
C. `7`
D. `11`
E. `15`
F. `True`
G. `False`

12) `list(filter(int, (-1.2, -0.5, 9.0, 0.2)))`
A. `[-0.5, 0.2]`
B. `[-1.2, 9.0]`
C. `[1, 0, 1, 0]`
D. `[-1.2, -0.5, 9.0, 0.2]`
E. `[True, False, True, False]`

13) `(lambda x: x + [9])([1])`
A. `[10]`
B. `[19]`
C. `[1, 9]`
D. `['10']`
E. `['19']`

14) `(lambda f: lambda x: f(f(f(x))))(lambda x: x + 2)(3)`
A. `7`
B. `9`
C. `11`
D. `24`
E. Some lambda function

15) `(lambda a, b, c: b(a(c)))(lambda x: x + 4, lambda x: x * 3, 1)`
A. `7`
B. `15`
C. `16`
D. Some function
E. Evaluating this expression yields an error

16) `(lambda x, y: y(x))(lambda x: x + 2, (lambda x: lambda y: x(x(y))))(3)`
A. `3`
B. `5`
C. `7`
D. `9`
E. `12`

17) `sum(map(lambda x: x(2),[lambda x: i + x for i in range(3)]))`
A. `3`
B. `6`
C. `9`
D. `12`
E. `15`

(proceed to the next page...)

18) What is the closest Python (built-in or packages) function to the following?

```
lambda f, l: [i for i in l if f(i)]
```

A. `filter`
B. `map`
C. `reduce`
D. `any`
E. `all`

Program Tracing [23 marks total]

For **questions 19 through 24**, you are given a complete Python program stored in a `.py` file. Determine the output of the program and choose the right option.

19) [2 marks]

```
x = 5
ans = 0
while x > 3:
    ans += x
    x -= 1
print(ans)
```

A. 4
B. 5
C. 9
D. 12
E. 15
F. 16

20) [3 marks]

```
q = 75
if q > 10:
    if q < 7:
        print('a')
    elif q > 75:
        print('b')
    else:
        print('c')
else:
    print('d')
```

A. a
B. b
C. c
D. d
E. Program execution completes successfully but prints nothing

21) [3 marks]

```
def f1(x):
    return f2(x) + f3(x)
def f2(x):
    return f3(x) + 2
def f3(x):
    return 3 + x
print(f1(1))
```

A. 10
B. 6
C. 5
D. 4
E. The program runs in an infinite loop
F. The program crashes and raises an exception

(proceed to the next page...)

22) [3 marks]

```
def p():
    return lambda: print('P!')
if not p:
    p()
elif p:
    p()()
else:
    print(not p)
```

A. `True`
B. `False`
C. `P!`
D. `None`
E. Some function

23) [3 marks]

```
a = {1: 2, 2: 4, 3: 6, 4: 7}
for k in a:
    if k%2 == 1:
        del a[k]
print(a)
```

A. `{1: 2, 2: 4, 3: 6, 4: 7}`
B. `{2: 4, 4: 7}`
C. `{}`
D. `{2: 4}`
E. `{1: 2, 2: 4, 3: 6}`
F. `{2: 4, 3: 6, 4: 7}`
G. The program crashes and raises an exception

24) [3 marks]

```
d = {0: 9, 1: 0, 2: 1, 3: 4, 4: 1, 5: 9, 6: 1, 0: 7, 4: 5}
a = 4
while a in d:
    a = d[a]
print(a)
```

A. 1
B. 7
C. 9
D. The program runs in an infinite loop
E. Executing this program yields an error

(proceed to the next page...)

For **questions 25 and 26**, you are given a complete Python program stored in a `.py` file. Determine the output of the program.

25) [3 marks]

```
def f(x,n):
    if n == 0:
        return x
    return 3 + g(x, n - 1)

def g(x,n):
    if n == 0:
        return x
    return 1 + f(x, n - 1)

print(f(2, 100))
```

Your Answer: _____

26) [3 marks]

```
ans = 0
l = [1, 2, 3, 4, 5]
for i in l:
    l.pop(0)
    ans += i
print(ans)
```

Your Answer: _____

Programming

Note in this section that no packages are imported.

27) [6 marks]. The `bin_search` function is an iterative implementation of binary search on an input list of integers:

```
>>> bin_search([i for i in range(10)], 3)
True
>>> bin_search([i * 2 for i in range(10)], 17)
False
```

An incomplete implementation of `bin_search` is given below. Fill in the blanks to complete it.

```
def binary_search(ls, key):
    start = <BLANK_1>
    end = <BLANK_2>
    m = <BLANK_3>
    while start <= end:
        if key == ls[m]:
            return True
        if key < ls[m]:
            end = <BLANK_4>
        if key > ls[m]:
            start = <BLANK_5>
        m = <BLANK_6>
    return False
```

Blank	Your Answer
<BLANK_1>	
<BLANK_2>	
<BLANK_3>	
<BLANK_4>	
<BLANK_5>	
<BLANK_6>	

(proceed to the next page...)

28) [4 marks]. The `deep_sum` function deeply sums an arbitrarily deeply nested list of integers:

```
>>> deep_sum([1, 2, [3], [4, 5, [6, 7],[]]])
28
```

An incomplete implementation of `deep_sum` is given below. Fill in the blanks to complete it.

```
def deep_sum(lst):
    if not lst:
        return 0
    if type(lst[0]) == list:
        return <BLANK_7>
    else:
        return <BLANK_8>
```

Blank	Your Answer
<BLANK_7>	
<BLANK_8>	

29) [5 marks]. You are given functions `Z` and `r`:

```
Z = (lambda x:(lambda f: f(f))(lambda y: x(lambda z: y(y)(z))))
r = lambda a: lambda b: lambda c: Z(lambda f: lambda x: a if b(x) else c(f, x))
```

Write the `factorial` function as a single-line expression making use of `r`:

```
factorial = _____________________________________________________
```

30) [3 marks]. If the input to the function `f` is a list of integers, what does `f` do?

```
def f(lst):
    lst2 = [(lst[i], lst[i + 1]) for i in range(len(lst) - 1)]
    for tup in lst2:
        if tup[0] > tup[1]:
            return False
    return True
```

A. To check if `lst` is non-decreasing (for each integer `i` in the list, the next integer is not smaller than `i`).
B. To check if `lst` is non-increasing (for each integer `i` in the list, the next integer is not larger than `i`).
C. To check if `lst` is strictly increasing (for each integer `i` in the list, the next integer is greater than `i`).
D. To check if `lst` is strictly decreasing (for each integer `i` in the list, the next integer is smaller than `i`).

(proceed to the next page...)

31) [8 marks]. You are the distributor of some products. Each product has a value and a weight, and they are given as a tuple—for example, if a particular Product A has a value of $10 and weighs 5 units, it will be represented as `(10, 5)`. Your company has at least one type of product, and has unlimited quantity of each product.

A buyer wants to buy as many products from you as possible. However, you only have one truck to deliver your products, it can only make a single trip, and it has a known limited weight capacity. Based on the product selection and weight capacity of the truck, you are to determine the maximum total value of products you can deliver with your truck in one trip.

As an example, if you have two types of products `[(10, 5), (7, 3)]` and the truck's capacity is `11`, then the best value will be $10+2(7)=24$ as you can load one unit of the first product and two units of the second product with a total weight of $5+2(3)=11$. However, if the truck's capacity is `12`, then you can transport a total value of $4(7)=28$ instead. Note that it is possible that the truck's capacity is not fully utilized.

The function `total_value(products, capacity)` computes the maximum total value of products you can deliver with the truck in one trip:

```
>>> total_value([(10, 5), (7, 3)], 11)
24
>>> total_value([(10, 5), (7, 3)], 12)
28
>>> total_value([(5, 2)], 11) # capacity is not full
25
```

An incomplete implementation of `total_value` is given below. Fill in the blanks to complete it.

```
def total_value(products, capacity):
    if products == []:
        return <BLANK_9>
    if capacity < products[0][1]:
        return <BLANK_10>
    a = products[0][0] + <BLANK_11>
    b = <BLANK_12>
    return max(a, b)
```

Blank	Your Answer
<BLANK_9>	
<BLANK_10>	
<BLANK_11>	
<BLANK_12>	

(proceed to the next page...)

Questions 32 through 34 refer to the same `count_sum` function.

32) [5 marks]. The `count_sum` function receives a list of unique integers and an integer target t, and returns the number of pairs of integers in the list who sum to t. Note that an integer cannot be used more than once to form any pair:

```
>>> count_sum([3, 5, 6, 4, 7], 9) # (3, 6), (5, 4)
2
>>> count_sum([3, 5, 6, 4, 7], 8) # (3, 5)
1
```

Fill in the blanks to complete the code:

```
def count_sum(lst, t):
    count = 0
    for i in range(<BLANK_13>, <BLANK_14>):
        for j in range(<BLANK_15>, <BLANK_16>):
            if <BLANK_17>:
                count+=1
    return count
```

Blank	Your Answer
<BLANK_13>	
<BLANK_14>	
<BLANK_15>	
<BLANK_16>	
<BLANK_17>	

33) [3 marks]. `count_sum` is very inefficient; if the input list has more than 1000 elements, the function needs a few minutes to compute the answer. Why is it so slow?

34) [3 marks]. Suggest a better way to implement `count_sum` to improve its speed.

(proceed to the next page...)

35) [4 marks]. We know that when class `B` extends/inherits/is a subclass of class `A`, then the functionality in class `A` is made available to class `B`. This makes sense because when `B` is a subclass of `A`, we say that `B` is-a `A`, such as when we say a `MinimalAccount` is-a `BankAccount`.

Knowing this, Mallory thinks, "Oh, since a square is a rectangle, then my `Square` class should extend my `Rectangle` class!". Therefore, she wrote the following:

```
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height
    def get_area(self):
        return self.width * self.height
    def set_w_h(self, new_width, new_height):
        self.width = new_width
        self.height = new_height

class Square(Rectangle):
    def __init__(self, side_length):
        super().__init__(side_length, side_length)
```

Is Mallory's implementation correct? Give reasons to support your answer.

– End of Exam –

This page was intentionally left blank.

Exam Solutions

Expression Evaluation. 1) A; 2) A; 3) D; 4) A; 5) E; 6) E; 7) E; 8) A; 9) B; 10) A; 11) A; 12) B; 13) C; 14) B; 15) B; 16) C; 17) D; 18) A;

Program Tracing. 19) C; 20) C; 21) A; 22) C. 23) G; 24) C; 25) 202; 26) 9.

Code Comprehension

27)

Blank	Correct Answer
<BLANK_1>	`0`
<BLANK_2>	`len(ls) - 1`
<BLANK_3>	`(start + end) // 2`
<BLANK_4>	`m - 1`
<BLANK_5>	`m + 1`
<BLANK_6>	`(start + end) // 2`

28)

Blank	Correct Answer
<BLANK_7>	`deep_sum(lst[0]) + deep_sum(lst[1:])`
<BLANK_8>	`lst[0] + deep_sum(lst[1:])`

29) `factorial = r(1)(lambda x: x == 0)(lambda f, x: x * f(x - 1))`

30) A;

31)

Blank	Correct Answer
<BLANK_9>	`0`
<BLANK_10>	`total_value(products[1:], capacity)`
<BLANK_11>	`total_value(products, capacity - products[0][1])`
<BLANK_12>	`total_value(products[1:], capacity)`

32)

Blank	Correct Answer
<BLANK_13>	`0`
<BLANK_14>	`len(lst)`
<BLANK_15>	`i + 1`
<BLANK_16>	`len(lst)`
<BLANK_17>	`lst[i] + lst[j] == v`

33) Its time complexity is $O(N^2)$ where N is the number of elements of `lst`.

34) Here is an $O(N)$ solution:

```
def count_sum(lst, t):
    elements = set(lst)
    count = 0
    for e in elements:
        if (t - e) != e and (t - e) in elements:
            count += 1
    return count
```

35) Incorrect. A square **is not** a rectangle because we can set the width and height of a rectangle to different values, but we should not be able to do the same for squares:

```
>>> square = Square(1)
>>> square.set_w_h(2, 3) # now it is no longer a square!
>>> square.get_area()
6
```