IT5001—*Software Development Fundamentals*

AY21/22 Semester 2 Quiz 2

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 quiz.

This document contains **6 printed pages**.
You have **30 minutes** to complete the quiz.
There are **19 questions**.

Section	**Awarded**	**Maximum**
Expression Evaluation I		15
Expression Evaluation II		24
Program Tracing I		24
Program Tracing II		16
Programming		21
Total		100

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

Expression Evaluation I [3 marks each]

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

1) `1 - 2 + 3 * 5 - 4`
A. `-28`
B. `2`
C. `6`
D. `10`
E. `16`

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

3) `[1, 2, 3, 4, 5, 6][3:5][2]`
A. `[]`
B. `[2]`
C. `[2, 3, 4]`
D. `[4, 5]`
E. Evaluating this expression yields an error

4) `list('abc') + list(['k'] + ['z'])`
A. `['abckz']`
B. `['k', 'z']`
C. `['abc', 'k', 'z']`
D. `['a', 'b', 'c', 'kz']`
E. `['a', 'b', 'c', 'k', 'z']`

5) `[5, [3], [2, 3]][[2, [1]][1]][:[1, 2][1]]`
A. `[]`
B. `[2]`
C. `[3]`
D. `[2, 3]`
E. Evaluating this expression yields an error

Expression Evaluation II [4 marks each]

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

6) `str(4) * 3 + str(2) * 4 + str(1 * 2)`
A. `'1282'`
B. `'432412'`
C. `'3333442'`
D. `'44422222'`
E. `'444222211'`

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

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

(proceed to the next page...)

9) `(lambda a, b, x, y: a(b(x, y)))((lambda a, b: a + b), (lambda a: a + 1), 5, 1)`
A. `7`
B. `11`
C. `(5, 1)`
D. Some function
E. Evaluating this expression yields an error

10) `'12345'[1:9][int('3') - 1][0][0]`
A. `[]`
B. `'3'`
C. `'4'`
D. `'5'`
E. Evaluating this expression yields an error

11)
```
# Assumming no package is imported:
(sqrt(-1)) or True
```
A. `True`
B. `False`
C. `None`
D. `1j`
E. Evaluating this expression yields an error

Program Tracing I [6 marks each]

In each of the following questions in this section, you are given a complete Python program stored in a `.py` file. Determine the output (if any) of the program upon execution, and choose the correct option.

12)

```
x = 1
for i in range(5):
    for j in range(2, 4):
        x = x + 2
print(x)
```

A. 10
B. 11
C. 16
D. 17
E. 20
F. 21

13)

```
q = 11
if q > 10:
    if q < 7:
        print('a')
    elif q > 9:
        print('b')
    else:
        print('c')
else:
    print('d')
```

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

(proceed to the next page...)

14)

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

A. 5
B. 7
C. 8
D. The program runs in an infinite loop
E. Executing this program yields an error (including `RecursionError`)

15)

```
x = {'a', 'bc', 'de'}
y = {'b', 'de', 'a', 'b'}
print(x ^ y | x)
```

A. {}
B. {'b'}
C. {'de', 'a'}
D. {'bc', 'b'}
E. {'a', 'b', 'de', 'bc'}

Program Tracing II [8 marks each]

In each of the following questions in this section, you are given a complete Python program stored in a `.py` file. Determine the output (if any) of the program upon execution, and choose the correct option.

16)

```
d = {0: 9, 1: 0, 2: 1, 3: 4, 4: 1, 5: 9, 6: 1, 0: 7}
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...)

17)

```
x = [1, 2, 3]
def foo(lst, x):
    if not lst:
        return 1
    return foo(lst[1:], x) + x(lst[0])
print(foo(x, lambda x: 4 - x))
```

A. 4
B. 6
C. 7
D. [1, 2, 3]
E. [3, 2, 1]
F. Executing this program yields an error

Programming [3 marks each blank]

Note in this section that no packages are imported.

18) The `turn_odd` function receives a positive integer argument and turns all of its even digits to odd digits by adding 1 to them:

```
>>> turn_odd(222111)
333111
>>> turn_odd(1234567890)
1335577991
```

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

```
def turn_odd(n):
    if n == 0:
        return 0
    if n % 2 == 1:
        return <BLANK_1> * 10 + <BLANK_2>
    else:
        return <BLANK_3> * 10 + <BLANK_4>
```

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

(proceed to the next page...)

19) The `diff_pair` function receives a list of **unique** integers `lst`, and a nonnegative integer `n`, and returns the number of pairs of integers in `lst` whose difference is equal to `n`:

```
>>> lst = [75, 80, 90, 77, 88, 91, 60, 74, 73, 70, 55, 93, 59]
>>> diff_pair(lst, 10) # (70, 80), (80, 90) and (60, 70)
3
>>> diff_pair(lst, 14) # (77, 91), (88, 74), (60, 74) and (73, 59)
4
```

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

```
def diff_pair(lst, n):
    count = 0
    for i in range(len(lst)):
        for j in range(<BLANK_5>, len(lst)):
            if <BLANK_6> == n:
                <BLANK_7>
    return count
```

Blank	Your Answer
<BLANK_5>	
<BLANK_6>	
<BLANK_7>	

– End of Quiz –

This page was intentionally left blank.

Quiz Solutions

Expression Evaluation. 1) D; 2) B; 3) E; 4) E; 5) E; 6) D; 7) A; 8) D; 9) E; 10) C; 11) E.

Program Tracing. 12) F; 13) B; 14) A; 15) E; 16) B; 17) C.

Code Comprehension.

Blank	Correct Answer
<BLANK_1>	`turn_odd(n // 10)`
<BLANK_2>	`n % 10`
<BLANK_3>	`turn_odd(n // 10)`
<BLANK_4>	`n % 10 + 1`
<BLANK_5>	`i + 1`
<BLANK_6>	`abs(lst[i] - lst[j])`
<BLANK_7>	`count += 1`