

**NATIONAL UNIVERSITY OF SINGAPORE**  
Department of Computer Science, School of Computing  
**IT5001—Software Development Fundamentals**  
Academic Year 2021/2022, Semester 1

**Final Assessment Solutions Manual**

16 October 2021

**Time allowed:** 2 hours

---

**INSTRUCTIONS TO CANDIDATES** (please read carefully):

1. This is a **closed-book assessment**. You are allowed **ONE (1)** A4-sized reference sheet, double-sided, printed or written, and **ONE (1)** blank A4 paper for scratch.
2. You may use a non-programmable calculator. Use of any other electronic devices, including smart watches, is **NOT** allowed.
3. Do **NOT** open this document until you are told to do so.
4. This assessment paper comprises **TWENTY-SIX (26) questions** and **TWELVE (12) pages** including this cover page.
5. **Do NOT** write your name on any document you submit. Write your Student Number (starting with A) in the space below.
6. You may write in blue or black with a pencil or pen.
7. The total attainable score for this assessment is **100 marks**. You must complete all questions to score full marks. This assessment counts towards **20%** of your final grade.
8. You **cannot** communicate with anyone other than the invigilators throughout the exam.
9. **You must attempt the assessment on your own.** The University takes a zero-tolerance approach towards plagiarism and cheating.

**Student Number:**

A								
---	--	--	--	--	--	--	--	--

---

**For Examiner's Use Only**

Section	Marks	Remarks
Expression Evaluation [24 marks]		
Program Tracing [24 marks]		
Program Comprehension [18 marks]		
Programming [34 marks]		
<b>Total</b> [100 marks]		

## Expression Evaluation [24 marks]

There are several questions in this section. Answer each question **independently and separately**.

In each question, one or more Python expressions are entered into a fresh Python shell with no prior import statements. Determine the result from evaluating the final expression entered and **write your answers at the end of the section**.

### Question 1) [2 marks]

`1 - 2 + 3 * 5 - 4`

Options:

- A. -28
- B. 2
- C. 6
- D. 10
- E. 16

### Question 3) [2 marks]

`True or False and False`

Options:

- A. True
- B. False
- C. 0
- D. None
- E. Evaluating this expression yields an error

### Question 5) [2 marks]

`(sqrt(-1)) or True`

Options:

- A. True
- B. False
- C. 1j
- D. None
- E. Evaluating this expression yields an error

### Question 7) [2 marks]

`list('abc') + ['k'] + ['z']`

Options:

- A. ['a', 'b', 'c', 'k', 'z']
- B. ['abc', 'k', 'z']
- C. ['a', 'b', 'c', 'kz']
- D. ['k', 'z']
- E. ['abckz']

### Question 2) [2 marks]

`(False == True) or False`

Options:

- A. True
- B. False
- C. 0
- D. None
- E. Evaluating this expression yields an error

### Question 4) [2 marks]

`'abcde'[2:5][1][0][0]`

Options:

- A. ''
- B. []
- C. 'd'
- D. 'cde'
- E. Evaluating this expression yields an error

### Question 6) [2 marks]

`False == True == False`

Options:

- A. True
- B. False
- C. 1
- D. None
- E. Evaluating this expression yields an error

### Question 8) [2 marks]

`list(['abc']) + list(('k', 'z'))`

Options:

- A. ['a', 'b', 'c', 'k', 'z']
- B. ['abc', 'k', 'z']
- C. ['a', 'b', 'c', 'kz']
- D. ['k', 'z']
- E. ['abckz']

(The **Expression Evaluation** section continues in the next page...)

**Question 9)** [2 marks]

```
(lambda x: x + [9])(['1'])
```

Options:

- A. [10]
- B. ['19']
- C. ['1', '9']
- D. [1, 9]
- E. ['1', 9]

**Question 10)** [2 marks]

```
f = lambda a, b: lambda x: b(b(a))
```

```
f('b', lambda a: a * 3)(lambda a: a[:1])
```

Options:

- A. Evaluating this expression yields some function
- B. 'b'
- C. 'bbb'
- D. 'bbbbbbbbbb'
- E. Evaluating this expression yields an error

**Question 11)** [2 marks]

```
[5, [3], [2, 3]][[2, [1]][1]][:[1, 2][1]]
```

Options:

- A. []
- B. [2]
- C. [3]
- D. [2, 3]
- E. Evaluating this expression yields an error

**Question 12)** [2 marks]

```
[5, [3], [2, 3]][[2, 1][0]][:[1, 2][1]]
```

Options:

- A. []
- B. [2]
- C. [3]
- D. [2, 3]
- E. Evaluating this expression yields an error

Write your answers for questions 1 to 12 in the space below:

Q1)	Q2)	Q3)	Q4)	Q5)	Q6)
Q7)	Q8)	Q9)	Q10)	Q11)	Q12)

(The next section begins in the next page...)

## Program Tracing [24 marks]

There are several questions in this section. Answer each question **independently and separately**.

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 **write your answers at the end of the section**.

### Question 13) [4 marks]

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

Options:

- A. 10
- B. 16
- C. 17
- D. 20
- E. 21

### Question 15) [4 marks]

```
1 def f1(x):
2     return '1' + f3(x)
3 def f2(x):
4     return f4(x) + '2'
5 def f3(x):
6     return f2(x) + '3'
7 def f4(x):
8     return '4' + str(x)
9 print(f2('0'))
```

Options:

- A. 402
- B. 3402
- C. 3042
- D. 13042
- E. 12340

### Question 14) [4 marks]

```
1 q = 11
2 if q > 10:
3     if q < 7:
4         print('a')
5     elif q > 9:
6         print('b')
7     else:
8         print('c')
9 else:
10    print('d')
```

Options:

- A. a
- B. b
- C. c
- D. d
- E. Executing this program does not produce any output

### Question 16) [4 marks]

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

Options:

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

**Question 17)** [4 marks]

```

1 lst1 = ['bc', 'de', 'ya',
2         'ab', 'bq', 'bd']
3 lst2 = []
4 for x in lst1:
5     lst2.append(tuple(x))
6 d = dict(lst2)
7 print(d['b'])

```

Options:

- A. a
- B. b
- C. c
- D. d
- E. Only a blank line is produced as output

**Question 18)** [4 marks]

```

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

```

Options:

- A. set()
- B. {'b'}
- C. {'bc', 'b'}
- D. {'de', 'a'}
- E. {'a', 'b', 'de', 'bc'}

Write your answers for questions 13 to 18 in the space below:

Q13)	Q14)	Q15)	Q16)	Q17)	Q18)
------	------	------	------	------	------

**Program Comprehension [18 marks]**

There are several questions in this section. Answer each question **independently and separately**.

In each of the following questions in this section, you are given a complete Python program stored in a .py file. Answer the questions posed to you and **write your answers in the empty space below the corresponding question**.

**Question 19)** [4 marks]. Observe the following program fragment, paying attention to function f19:

```

1 def f19(lst):
2     while not boo(lst, 0):
3         i = randint(0, len(lst) - 1)
4         lst[i], lst[0] = lst[0], lst[i]
5     return lst
6
7 def boo(lst, N):
8     l1 = lst[1:]
9     l2 = []
10    for i in range(len(l1)):
11        l2.append(l1[i] - lst[i])
12    return min(l2) >= N

```

Assuming the argument to f19 is a list of integers whose length exceeds 1, **describe function f19; or in other words, what does f19 do?**

**Question 20)** [4 marks]. Observe the following program:

```
1 class RPGCharacter:
2     def __init__(self):
3         self.action = 'Doing Nothing'
4     def act(self):
5         print(self.action)
6
7 class Fighter(RPGCharacter):
8     def __init__(self):
9         self.action = 'Fight'
10
11 class Cleric(RPGCharacter):
12     def __init__(self):
13         self.action = 'Heal'
14
15 class Paladin(Fighter,Cleric):
16     pass
17
18 me = Paladin()
19 me.act()
```

**What is the output of this program?**

**Question 21)** [4 marks]. Observe the following program.

```
1  # Mage if job == 0
2  # Warrior if job == 1
3  class RPGCharacter:
4      def __init__(self,name,job):
5          self.name = name
6          self.job = job
7          self.title = ['the Mage', 'the Warrior'][job]
8
9      def action(self):
10         print(self.name + " " + self.title + " attacks!")
11
12     def changeJob(self,job):
13         self.job = job
14
15 me = RPGCharacter("Gandalf", 1)
16 me.changeJob(0)
17 me.action()
```

**What is the output of this program?**

**Question 22)** [6 marks]. Observe the following program fragment.

```
1  def f22(a, b):
2      ans = 0
3      while b > 1:
4          if b % 2:
5              ans += a
6              b -= 1
7          else:
8              a += a
9              b //= 2
10     return ans + a
```

Assuming the arguments to `f22` are both strictly positive integers, **describe function `f22` and explain the significance of implementing `f22` in this way.**

## Programming [34 marks]

This section contains multiple questions. Answer each question **independently and separately**.

In each question, you are given an incomplete Python program stored in a `.py` file. Answer the questions posed to you and write your answers at the end of this section by replacing each blank with a syntactically correct Python expression/statement. You can only obtain full marks for this question if you answer accurately, concisely, and write legibly.

**Question 23)** [6 marks]. Given a list `ls` of unique integers, the function `diff_pair(ls, n)` will determine how many pairs of numbers in `ls` differ by `n`.

Example uses of `diff_pair` follow:

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

An incomplete implementation of `diff_pair` is given below. Replace each blank with a valid Python expression/statement and write your answers in the space below.



```

1 def diff_pair(L,N):
2     a = len(L)
3     count = 0
4     for i in range(0,a):
5         for j in range(<BLANK_1>, a):
6             if <BLANK_2> == N:
7                 count += 1
8     return count

```

Blank	Your Answer
<BLANK_1>	
<BLANK_2>	

**Question 24)** [9 marks]. You are given a map like our Assignment below:

```

WWWWWWWWWWWWWWWWWWWW.TTT..^^^..WWWWWWWW
WWWWWWWWWWWWWWWWWWWW..T..^^^..WWWWWWWW
WWWWWWWWWWWWWWWWWWWW..T.T..^^^..WWWWWWWW
WWWWWWWWWWWWWWWWWWWW..T....^^^..T.WWWWWWW
WWWWWWWWWWWWWWWWWWWW.....^^^..T..WWWWWW
WWWWWWWWWWWWWWWWWWWW...T...^^^..T.T.WWWWWWW
WWWWWWWWWWWWWWWWWWWW..TTT..^^^..WWWWWWWW
WWWWWWWWWWWWWWWWWWWW..T..^^^..WWWWWWWW
WWWWWWWWWWWWWWWWWWWW.....^.....WWWWWWWW
WWWWWWWWWWWWWWWWWWWW..T.T.....WWWWWWWW
WWWWWWWWWWWWWWWWWWWW...T.....WWWWWWWWWW
WWWW..WWWWWWWWWWWW..T.T.....WWWWWWWWWW
WWWW..T.T.WWWWWWWWWWW.....WWWWWWWWWWWW
WWWWWWWW..T..WWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWW.T.WWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWW..WWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWW..T.T..WWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWW.T.T..WWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWW.T..WWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWW..WWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW

```

And the difference is that it is given in a 2D array of any size with  $r$  rows and  $c$  columns. Each entry in the array will be either a character 'W' (water), 'T' (Tree), '.' (Land) or '^' (hills).

There are treasures buried in different locations, and X marks the spot! It means that the treasure location is in the middle of 5 trees that form an X shape. (Sometimes, it is allowed to have more than 5 trees.) Please see the red tree in the above map that shows where are the treasures. For that map, there are four treasures.

The function `count_treasure(t_map)` returns the number of treasures hidden in the map.

An incomplete implementation of `count_treasure` is given below. Replace each blank with a valid Python expression/statement and write your answers in the space below.

```

1 def count_treasure(t_map):
2     count = 0
3     r = len(t_map)
4     c = len(t_map[0])
5     for i in range(<BLANK_3>):
6         for j in range(<BLANK_4>):
7             if <BLANK_5>:
8                 count += 1
9     return count

```

Blank	Your Answer
<BLANK_3>	
<BLANK_4>	
<BLANK_5>	

**Question 25)** [9 marks]. The `deep_concatenate` function receives an arbitrarily deeply nested list of strings, and returns the concatenation of all the strings:

```

>>> deep_concatenate(['a', 'b', ['c', 'd', ['e', 'f'], 'g'], [], 'h'])
'abcdefgh'

```

An incomplete implementation of `deep_concatenate` is given below. Replace each blank with a valid Python expression/statement and write your answers in the space below.

```

1 def deep_concatenate(ls):
2     if not ls:
3         return <BLANK_6>
4     if type(ls) == <BLANK_7>:
5         return ls
6     return <BLANK_8> + deep_concatenate(ls[1:])

```

Blank	Your Answer
<BLANK_6>	
<BLANK_7>	
<BLANK_8>	

**Question 26)** [10 marks]. The `lcs` is supposed to compute the Longest Common Subsequence of two strings. A subsequence  $b$  of another sequence  $a$  is itself a sequence of elements, such that the elements of  $b$  appear in the same relative order in  $a$ , but need not necessarily be contiguous. For example, 'abc', 'abg', 'bdf', 'acefg' etc are subsequences of 'abcdefg'. 'acb' is not a subsequence of 'abcdefg' because 'c' appears before 'b' in 'acb' but not in 'abcdefg'.

`lcs` is supposed to work in this way:

```
>>> lcs('AGGTAB','GTXAYB') # GTAB is one of the longest common subsequences
4
```

The following program doesn't seem to work:

```
1 def lcs(X, Y):
2     def lcs_helper(X, Y, m, n):
3         if m == 0 or n == 0:
4             return 0;
5         if X[m - 1] == Y[n - 1]:
6             ans = 1 + lcs_helper(X, Y, m - 1, n - 1);
7         else:
8             ans = max(lcs_helper(X, Y, m, n - 1),
9                       lcs_helper(X, Y, m - 1, n))
10        return ans
11    return lcs_helper(X, Y, len(X), len(Y))
```

**Question 26a)** [2 marks] What's wrong with the code above?

**Question 26b) [8 marks]** Amend the function implementation by replacing the blanks below with syntactically correct Python expressions/statements and write your answers in the space below.

```
1 def lcs(X,Y):
2     def lcs_helper(X, Y, m, n):
3         <BLANK_9>
4         <BLANK_10>
5         if m == 0 or n == 0:
6             return 0;
7         if X[m - 1] == Y[n - 1]:
8             ans = 1 + lcs_helper(X, Y, m - 1, n - 1);
9         else:
10            ans = max(lcs_helper(X, Y, m, n - 1),
11                      lcs_helper(X, Y, m - 1, n))
12            <BLANK_11>
13            return ans
14    <BLANK_12>
15    return lcs_helper(X, Y, len(X), len(Y))
```

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

**– End of Assessment –**

## Solutions

Q1) D	Q2) B	Q3) A	Q4) C	Q5) E	Q6) B
Q7) A	Q8) B	Q9) E	Q10) D	Q11) E	Q12) D
Q13) E	Q14) B	Q15) A	Q16) C	Q17) D	Q18) E

19) The function returns the equivalent of the list that is sorted in ascending order.

20) Fight

21) Gandalf the Warrior attacks!

22) Returns  $a \times b$  in  $O(\log b)$  time.

26a) It is slow.

Blank	Answer
<BLANK_1>	<code>i + 1</code>
<BLANK_2>	<code>abs(L[i] - L[j])</code>
<BLANK_3>	<code>1, r - 1</code>
<BLANK_4>	<code>1, c - 1</code>
<BLANK_5>	<code>t_map[i][j] == t_map[i-1][j-1] == t_map[i+1][j-1] == t_map[i+1][j+1] == t_map[i-1][j+1] == 'T'</code>
<BLANK_6>	<code>''</code>
<BLANK_7>	<code>str</code>
<BLANK_8>	<code>deep_concatenate(ls[0])</code>
<BLANK_9>	<code>if (n, m) in memo:</code>
<BLANK_10>	<code>return memo[n, m]</code>
<BLANK_11>	<code>memo[n, m] = ans</code>
<BLANK_12>	<code>memo = {}</code>