NATIONAL UNIVERSITY OF SINGAPORE

Department of Computer Science, School of Computing

IT5001—*Software Development Fundamentals*

Academic Year 2022/2023, Semester 2

Final Assessment

QUESTION BOOKLET

18 March 2023 **Time allowed:** 2 hours

INSTRUCTIONS TO CANDIDATES (please read carefully):

1. This is a **CLOSED-BOOK assessment**. You are only allowed **ONE (1)** A4-sized reference sheet, double-sided, printed or written, and **ONE (1)** additional blank A4-sized paper for scratch.
2. You may use a non-programmable calculator.
3. Use of any other electronic devices, including smartwatches, is **NOT** allowed.
4. The assessment consists of TWO documents —the 'QUESTION BOOKLET' (this document) and the 'ANSWER BOOKLET'. **Do NOT open these documents until you are told to do so**.
5. This 'QUESTION BOOKLET' comprises **THIRTY-SEVEN (37) questions** and **ELEVEN (11) pages** including this cover page.
6. The 'ANSWER BOOKLET' comprises **SIX (6) pages** including the cover page.
7. **Do NOT write your name anywhere in the 'ANSWER BOOKLET'**.
8. Write and shade your Student Number (starting with `A`) in the 'ANSWER BOOKLET'.
9. Write or shade all your answers in the 'ANSWER BOOKLET'.
10. **Shade each bubble completely with a pencil (at least 2B). You may write with a pencil (at least 2B) or pen (no red ink)**.
11. **All your answers must be written in the 'ANSWER BOOKLET'**. No extra sheets will be accepted as answers. You may use this 'QUESTION BOOKLET' as scratch paper. **You are to submit both booklets at the end of the assessment**.
12. The total attainable score for this assessment is **100 marks**. You must complete all questions to score full marks. This assessment counts towards **40%** of your final grade.
13. You **cannot** communicate with anyone other than the invigilators throughout the exam.
14. **You must attempt the assessment on your own**. The University takes a zero-tolerance approach towards plagiarism and cheating.

Expression Evaluation [16 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 last expression entered and shade the correct option in the 'ANSWER BOOKLET' with a **pencil (at least 2B)**.

Question 1) [1 mark]
`max(-1, 2, -3)`
Options:
A. `-1`
B. `2`
C. `-3`
D. `None`
E. Evaluating this expression yields an error

Question 2) [1 mark]
`6 // 4 + 1`
Options:
A. `2`
B. `2.0`
C. `2.5`
D. `None`
E. Evaluating this expression yields an error

Question 3) [1 mark]
`'abcde'[-1:7]`
Options:
A. `''`
B. `'e'`
C. `'edcba'`
D. `None`
E. Evaluating this expression yields an error

Question 4) [1 mark]
`bool('False')`
Options:
A. `True`
B. `False`
C. `None`
D. Evaluating this expression yields an error
E. None of the above

Question 5) [1 mark]
`'' in 'abc'`
Options:
A. `True`
B. `False`
C. `0`
D. `None`
E. Evaluating this expression yields an error

Question 6) [1 mark]
`[1, 3] * 2`
Options:
A. `[2, 6]`
B. `[1, 3, 2]`
C. `[1, 3, 1, 3]`
D. `None`
E. Evaluating this expression yields an error

Question 7) [1 mark]
`(1, [2], 3)[1]`
Options:
A. `[1]`
B. `[2]`
C. `[3]`
D. `1`
E. Evaluating this expression yields an error

Question 8) [1 mark]
`[[1, 2], [3, 4, 5]][1][2]`
Options:
A. `5`
B. `[1, 2]`
C. `[3, 4, 5]`
D. `None`
E. Evaluating this expression yields an error

Question 9) [1 mark]
`sorted('abracadabra')[:3]`
Options:
A. `'aaa'`
B. `['a', 'a', 'a']`
C. `['a', 'b', 'r']`
D. `['abracadabra']`
E. Evaluating this expression yields an error

Question 10) [1 mark]
`{1: 2}[1] + {3: 4}[3]`
Options:
A. `4`
B. `5`
C. `6`
D. `None`
E. Evaluating this expression yields an error

Question 11) [1 mark]
`{1: 2, 3: 4}.get(2)`
Options:
A. `1`
B. `2`
C. `3`
D. `None`
E. Evaluating this expression yields an error

Question 12) [1 mark]
`{1: {2: {3: 4}}}[{1: 2, 3: 4}[1]]`
Options:
A. `1`
B. `{3: 4}`
C. `{2: {3: 4}}`
D. `None`
E. Evaluating this expression yields an error

Question 13) [1 mark]
`list(map(lambda i: i - 1, [1, 2]))`
Options:
A. `[0, 1]`
B. `[1, 2]`
C. `<map object at 0x123456>`
D. `None`
E. Evaluating this expression yields an error

Question 14) [1 mark]
`list(filter(lambda i: i - 1, [0, 1, 2]))`
Options:
A. `[0, 2]`
B. `[1, 2]`
C. `[-1, 1]`
D. `None`
E. Evaluating this expression yields an error

Question 15) [1 mark]
```
from functools import reduce
reduce(max, map(abs, [-3, 2, 4, -6]))
```
Options:
A. `3`
B. `4`
C. `6`
D. `None`
E. Evaluating this expression yields an error

Question 16) [1 mark]
```
a = map(int, '123')
max(a) + min(a)
```
Options:
A. `'31'`
B. `4`
C. `[3, 1]`
D. `None`
E. Evaluating this expression yields an error

Multiple Statement Questions [16 marks]

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

In each of these questions you are given a statement and offered several options. For each of these, choose the most appropriate option and shade the correct option in the 'ANSWER BOOKLET' with a **pencil (at least 2B)**. Note that the line numbers to the left of each code block are not part of the program.

Question 17) [2 marks]. Observe the following code snippet and some remarks about it.

```
1 a = int(input())
2 cond = a == 1
3 if cond == True:
4     print(1)
5 else:
6     print('not 1')
```

1. Line 3 can be replaced with "`if cond:`" and the code snippet would behave identically
2. Upon execution of this snippet, a `TypeError` will be raised from line 4 because the `print` function accepts strings only
3. Upon execution of this snippet, a `ValueError` will be raised from line 1 if the user enters a floating point number into the console

Which of the remarks is/are true?

A. Only 1 is correct
B. Only 2 is correct
C. Only 3 is correct
D. 1 and 2 are correct, but not 3
E. 1 and 3 are correct, but not 2

Question 18) [2 marks]. Bob wrote the following code snippet in hopes of letting `tp` be the tuple `(1, 2, 3, 4, 5)`:

```
1 tp = ()
2 for i in range(5):
3     tp += i
4 print(tp) # should be (1, 2, 3, 4, 5)
```

What can you say to Bob to convince him that his code can be better?

A. Line 3 causes a `TypeError`
B. Repeated concatenation makes his code snippet run in $O(n^2)$ time (where $n = 5$), which is suboptimal since there is an $O(n)$ approach
C. His comment in line 4 (`# should be (1, 2, 3, 4, 5)`) wrongly describes the contents of `tp`
D. More than one of the above; Bob has a lot to work on
E. None of the above; Bob's code is perfect

Question 19) [2 marks]. Which of the following statements is true of lists and tuples?

A. Lists are immutable (cannot be mutated), tuples are mutable (can be mutated)
B. Where `tp` is a tuple, an operation like `tp += (1,)` implicitly calls the `extend` method of tuples
C. `len(([1, 2, (3, 4)],))` evaluates to `4` because the length of a list/tuple is the sum of the lengths of its elements (the length of atomic objects like integers is 1)
D. More than one of the above are true
E. None of the above is true

Question 20) [2 marks]. Recount this recursive implementation of `fib` that computes the fibonacci numbers:

```
def fib(n):
    if n == 0 or n == 1: return 1
    return fib(n - 1) + fib(n - 2)
```

Which of the following statements is true of `fib`?

A. Memoizing this function using a list is just as (if not more) efficient than memoizing this function using a dictionary
B. Infinite recursion will occur when `fib(-1)` is called
C. No implementation of `fib` using loops can outperform this recursive implementation by any metric
D. More than one of the above are true
E. None of the above is true

Question 21) [2 marks]. Observe this implementation of `sum_terms`:

```
def sum_terms(term, seq):
    return sum(term(i) for i in seq)
```

Alice argues that `sum_terms` can be used for at least three purposes:

1. Computing the price of a customizable burger
2. Computing the approximation of `sin` up to n terms in its Taylor Series expansion, for some n
3. Counting the number of prime numbers from 2 to n, for some n

Which of Alice's arguments are correct?

A. Only 1 is correct
B. Only 2 is correct
C. Only 3 is correct
D. 1 and 2 are correct, but not 3
E. 1, 2 and 3 are all correct

Question 22) [2 marks]. Observe the `Duck` class:

```
class Duck:
    def __init__(self):
        self.sound = 'quack'
```

Which of the following is true of `Duck`?

A. Instances of `Duck` are actually dictionaries; therefore `Duck()['sound']` would evaluate to `'quack'`
B. If we define a class `BrownDuck` that inherits `Duck`, then all instances of `BrownDuck` will also have an attribute called `sound`
C. The expression `Duck().sound` will always evaluate to `'quack'`
D. More than one of the above are true
E. None of the above is true

Question 23) [2 marks]. Which of the following is true of exceptions?

A. Raising exceptions can be useful for detecting errors
B. Developers should never intentionally raise exceptions because it could crash the program
C. `finally` clauses in a `try-except` block are never needed
D. More than one of the above are true
E. None of the above is true

Question 24) [2 marks]. Among the following sorting algorithms, which is the fastest asymptotically (i.e. order-of-growth is the smallest)?

A. Bogosort (randomly shuffling the list until it becomes sorted)
B. Bubblesort (perform $n-1$ iterations of bubbling, each iteration of bubble performs swaps on each successive unsorted pair)
C. Mergesort (Divide list into two, recursively mergesort each half, merge the two sorted lists into one sorted list)
D. Exactly two of the above are tied to be, asymptotically, the fastest algorithms
E. All three of the above are, asymptotically, equally fast

Program Tracing [16 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 shade the correct option in the 'ANSWER BOOKLET' with a **pencil (at least 2B)**.

Question 25) [2 marks]

```
1 def f25(n):
2     if n > 5: return 5
3     if n > 3: return 3
4     if n > 1: return 1
5     return 0
6 print(f25(2))
```

Options:
A. `(1, 0)`
B. `0`
C. `3`
D. The program does not terminate
E. None of the above

Question 26) [2 marks]

```
1 def f26(seq):
2     if isinstance(seq, str):
3         return seq
4     return ''.join([f26(i) for i in seq])
5 print(f26(['a', ['b', ['c']], [['d'], 'e']]))
```

Options:
A. `['a', ['b', ['c']], [['d'], 'e']]`
B. `['e', 'd', 'c', 'b', 'a']`
C. `abcde`
D. The program does not terminate
E. None of the above

Question 27) [2 marks]

```
def f27(d):
    acc = {}
    for k, v in d.items():
        if v not in acc:
            acc[v] = []
        acc[v].append(k)
    return acc
print(f27({1: 2, 2: 3, 3: 2, 4: 2, 5: 4}))
```

Options:
A. `[1, 2, 3, 4, 5]`
B. `{2: 4, 3: 2, 4: 5}`
C. `{2: [1, 3, 4], 3: [2], 4: [5]}`
D. The program crashes with a `KeyError`
E. None of the above

Question 28) [2 marks]

```
def f28a(f, ls):
    if not ls: return ls
    return f(ls[0])
def f28b(s1):
    def g(s2):
        if s2.isalpha(): return [s1 + s2]
        return []
    return g
print(f28a(f28b('b'), f28a(f28b('0'), ['a'])))
```

Options:
A. `ab`
B. `b0a`
C. `['b', 'a']`
D. `[]`
E. None of the above

Question 29) [2 marks]

```
from functools import reduce
_ = map(lambda i: i + '2', '123')
_ = map(int, _)
_ = filter(lambda i: i % 4 == 0, _)
x = reduce(lambda i, j: i - j, _)
print(x)
```

Options:
A. `-20`
B. `0`
C. `12`
D. `1232`
E. None of the above

Question 30) [2 marks]

```
def f30():
    ls = []
    try:
        ls.append(1)
        ls.append(1 / 0)
    except:
        ls.append(2)
    finally:
        return ls
print(f30())
```

Options:
A. `[1]`
B. `[1, 2]`
C. `[1, inf, 2]`
D. The program crashes with a `ZeroDivisionError`
E. None of the above

Question 31) [2 marks]

```
class C31:
    def __init__(self, x):
        self.x = x
    def f31(self, c):
        if self.x > c.x:
            return self
        return c
a = C31(1)
b = C31(2)
c = a.f31(b)
print(c.x)
```

Options:
A. `1`
B. `2`
C. `None`
D. The program does not terminate
E. None of the above

Question 32) [2 marks]

```
class A:
    def __init__(self, x):
        self.x = x
    def f32(self, a):
        self.x += a.x
class B(A):
    def __init__(self, y):
        self.y = y
a = A(1)
b = B(2)
b.f32(a)
print(b.x)
```

Options:
A. `1`
B. `2`
C. `3`
D. The program does not terminate
E. None of the above

Programming [52 marks]

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

Answer the questions posed to you and **write your answers in the space provided in the 'ANSWER BOOKLET'**. You can only obtain full marks for this question if you answer accurately, concisely, and write legibly. You may choose to write in pencil or pen.

Question 33) [8 marks]. There are n cities numbered $0, 1, \ldots, n-1$, arranged in a circle where for each i, city i is connected to cities $i-1$ and $i+1$ (cities 0 and $n-1$ are connected to each other). You are also given a list A consisting of n nonnegative integers, indicating that the cost of entering city i is $\$A[i]$.

The `cheapest` function receives list A and two cities x and y such that $0 \leq x, y < |A|$ ($|A|$ is the length of A), and as a result returns the least amount of money needed to travel from city x to y (the cost of x is not included in the total cost).

Example runs:

```
>>> A = [5, 4, 3, 4, 5]
>>> cheapest(A, 1, 4)
10 # 1 -> 0 -> 4
>>> cheapest(A, 0, 2)
7 # 0 -> 1 -> 2
>>> cheapest(A, 3, 3)
0 # 3
```

Template:

```
def cheapest(A, x, y):
    a = <BLANK_33A>
    b = <BLANK_33B>
    return min(a, b)
```

Example runs and an incomplete implementation of `cheapest` are given above. **Replace each blank with a valid Python expression/statement and write your answers in the 'ANSWER BOOKLET'.**

Question 34) [10 marks]. An arbitrarily deeply nested list can be hard to deal with since there may be unintentional mutations on them. Instead of expecting to perform deep copying all the time, we can simply convert such a list into a deeply nested tuple once, to ensure that the entire object and its contents are not mutated.

The `deep_tuple` function converts an arbitrarily deeply nested list consisting only of integers, strings, floating point numbers or booleans, into a tuple:

Example runs:

```
>>> deep_tuple([1, [2.0, ['3', [False]]]])
(1, (2.0, ('3', (False,))))
>>> deep_tuple([[[[1]]]])
((((1,),),),)
```

Template:

```
def deep_tuple(s):
    if not isinstance(s, list):
        return <BLANK_34A>
    return <BLANK_34B>
```

Example runs and an incomplete implementation of `deep_tuple` are given above. **Replace each blank with a valid Python expression/statement and write your answers in the 'ANSWER BOOKLET'.**

Question 35 [10 marks]. Observe the following program:

```
def read(filename):
    with open(filename, 'w+') as f:
        _ = f.read()
        _ = _.strip()
        _ = _.split('\n')
        _ = map(lambda s: s.split(','), _)
        _ = map(lambda r: [convert(i) for i in r], _)
        return list(_)
```

```
def convert(i):
    try: return int(i)
    except: pass
    try: return bool(i)
    except: pass
    try: return float(i)
    except: pass
    return i
```

The `read` function is supposed to read a CSV file and convert it into a usable 2D list. If `read` is defined correctly, if a CSV file called `data.csv` looks like the one below (left), then `read` should produce the 2D list shown below (right) (notice that numbers in the file are represented accurately as numbers in the 2D list, likewise for Booleans).

An example file called `data.csv`

```
S/N,Name,Present,Score
1,Alan,True,48.9
2,Yong Qi,False,0
3,Jeanette,True,36.5
```

How `read` *should behave*

```
>>> read('data.csv')
[['S/N', 'Name', 'Present', 'Score'],
 [1, 'Alan', True, 48.9],
 [2, 'Yong Qi', False, 0],
 [3, 'Jeanette', True, 36.5]]
```

However, in reality, when `read('data.csv')` is called, the output is `[[False]]`, and the contents of `data.csv` are wiped from the file.

Propose changes to the program so that the `read` **function behaves correctly.** You do not need to write code in your solution unless it aids your explanation. Your proposal may contain changes to `read`, changes to `convert`, and/or to include other functions which will help `read` operate correctly.

Question 36 [12 marks]. Observe the following program, where the `time` function from the `time` library returns the current time:

```
1  from time import time
2  def reverse(s):
3      return s[::-1]
4  def timed_reverse(s):
5      start_time = time()
6      z = reverse(s)
7      end_time = time()
8      print(end_time - start_time)
9      return z
10 def factorial(n):
11     return 1 if n == 0 else \
12            n * factorial(n - 1)
13 def timed_factorial(n):
14     start_time = time()
15     z = factorial(n)
16     end_time = time()
17     print(end_time - start_time)
18     return z
```

Notice that effectively, the `timed_reverse` and `timed_factorial` functions both share the same program logic—this is in violation of the Don't Repeat Yourself (DRY) principle.

Redefine the `timed_reverse` **and** `timed_factorial` **functions using sound programming practices**. You should not need to redefine `reverse` and `factorial`.

Question 37 [12 marks]. You were given these two classes in Assignment 7:

```
class Character(object):
    def __init__(self):
        self.name = ''
        self.maxhp = 1000
        self.hp = 1000
        self.str = 0
        self.maxmana = 0
        self.mana = 0
        self.cost = 9999999999
        self.alive = True

    def act(self, myTeam, enemy):
        pass

    def gotHurt(self, damage):
        if damage >= self.hp:
            self.hp = 0
            self.alive = False
        else:
            self.hp -= damage
```

```
class Fighter(Character):
    def __init__(self):
        super().__init__()
        self.name = 'Fighter'
        self.maxhp = 1200
        self.hp = 1200
        self.str = 100
        self.cost = 100

    def act(self, myTeam, enemy):
        target = randAlive(enemy)
        enemy[target].gotHurt(self.str)
```

The game now includes a new character, `Radeon`. `Radeon` is just like a `Fighter`, but has the following characteristics:

1. Just like `Fighters`, `Radeons` have 100 strength (`self.str`). However, `Radeons` cost 200 and only have a maximum HP (`self.maxhp`) of 900;
2. `Radeons` have an additional attribute `power` which starts at 0;
3. As you all know, when an enemy attacks a `Fighter` with x damage, then the `Fighter`'s HP is reduced by x. However, when an enemy attacks a `Radeon` with x damage, the `Radeon`'s HP is only reduced by $3x/4$ but its power is increased by $x/4$ (its power is increased even if it dies from the attack);
4. When it is a `Radeon`'s turn to act, if its power is below 500, then it will attack a randomly chosen enemy just like `Fighters`, but also increase its power by 100. Otherwise, if its power is at least 500, it will attack five randomly chosen enemies, depleting all its power. The damage that a `Radeon` deals to an enemy on each attack is also equivalent to its strength, just like `Fighters`.

Write the `Radeon` class using sound Object-Oriented principles and practices. Assume you have direct access to the `Character` and `Fighter` classes (i.e. your code will be appended right below the `Fighter` class); the `Character` and `Fighter` classes should not be re-defined in your solution. Further assume that functions supporting teams (such as `randAlive` or `randDeath`) are also provided.

– End of Assessment –

NATIONAL UNIVERSITY OF SINGAPORE

Department of Computer Science, School of Computing

IT5001—*Software Development Fundamentals*

Academic Year 2022/2023, Semester 2

Final Assessment

SOLUTIONS MANUAL

18 March 2023 **Time allowed:** 2 hours

Multiple-Choice Questions

	A	B	C	D	E		A	B	C	D	E		A	B	C	D	E
1	○	●	○	○	○	**2**	●	○	○	○	○	**3**	○	●	○	○	○
4	●	○	○	○	○	**5**	●	○	○	○	○	**6**	○	○	●	○	○
7	○	●	○	○	○	**8**	●	○	○	○	○	**9**	○	●	○	○	○
10	○	○	●	○	○	**11**	○	○	○	●	○	**12**	○	○	○	○	●
13	●	○	○	○	○	**14**	●	○	○	○	○	**15**	○	○	●	○	○
16	○	○	○	○	●	**17**	○	○	○	○	●	**18**	○	○	○	●	○
19	○	○	○	○	●	**20**	●	○	○	○	○	**21**	○	○	○	○	●
22	○	○	●	○	○	**23**	●	○	○	○	○	**24**	○	○	●	○	○
25	○	○	○	○	●	**26**	○	○	●	○	○	**27**	○	○	●	○	○
28	○	○	○	●	○	**29**	●	○	○	○	○	**30**	○	●	○	○	○
31	○	●	○	○	○	**32**	○	○	○	○	●						

Programming

Question 33) [8 marks]; just use sum

```
def cheapest(A, x, y):
    a = sum(A[min(x, y):max(x, y) + 1]) - A[x]
    b = sum(A[:min(x, y) + 1] + A[max(x, y):]) - A[x]
    return min(a, b)
```

Question 34) [10 marks]; similar to Q26

```
def deep_tuple(s):
    if not isinstance(s, list): return s
    return tuple(deep_tuple(i) for i in s)
```

Question 35) [10 marks]. Line 2 should be `with open(filename) as f:`

`convert` should look like the following:

```
1  def convert(i):
2      try: return int(i)
3      except: pass
4      try: return float(i)
5      except: pass
6      if i == 'True': return True
7      if i == 'False': return False
8      return i
```

Question 36) [12 marks]. Retain similarities and parameterize differences

```
1  def timed(f, x):
2      start_time = time()
3      z = f(x)
4      end_time = time()
5      print(end_time - start_time)
6      return z
7  timed_factorial = lambda n: timed(factorial, n)
8  timed_reverse = lambda s: timed(reverse, s)
```

Question 37) [12 marks].

```
1  class Radeon(Fighter):
2      def __init__(self):
3          super().__init__()
4          self.name = 'Radeon'
5          self.cost = 200
6          self.hp = self.maxhp = 900
7          self.power = 0
8      def gotHurt(self, damage):
9          super().gotHurt(3 * damage // 4)
10         self.power += damage // 4
11     def act(self, myTeam, enemy):
12         if self.power >= 500:
13             for _ in range(5):
14                 super().act(myTeam, enemy)
15             self.power = 0
16         else:
17             super().act(myTeam, enemy)
18             self.power += 100
```

– End of Solutions Manual –