NATIONAL UNIVERSITY OF SINGAPORE

Department of Computer Science, School of Computing

IT5001—*Software Development Fundamentals*

Academic Year 2022/2023, Semester 2

Mid-Term Assessment

QUESTION BOOKLET

11 February 2023 **Time allowed:** 1 hour

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, NUS-approved calculator.
3. Use of any other electronic devices, including smart watches, 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 **TWENTY-SIX (26) questions** and **SIX (6) pages** including this cover page.
6. The 'ANSWER BOOKLET' comprises **FOUR (4) 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/shade all your answers in the 'ANSWER BOOKLET'. **Shade each bubble completely with a pencil (at least 2B).** No extra sheets will be accepted as answers. **You may write with a pencil (at least 2B) or pen (no red ink)**.
10. **You are required to only submit the 'ANSWER BOOKLET' at the end of the assessment**. You may use this 'QUESTION BOOKLET' as scratch paper.
11. 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.
12. You **cannot** communicate with anyone other than the invigilators throughout the exam.
13. **You must attempt the assessment on your own**. The University takes a zero-tolerance approach towards plagiarism and cheating.

Expression Evaluation [24 marks]

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

In each question, a Python expression is entered into a fresh Python shell with no prior import statements. Determine the result from evaluating the expression entered and shade the correct option in the 'ANSWER BOOKLET' with a **pencil (at least 2B)**.

Question 1) [2 marks]

```
6 * 5 - 4 / 2
```

Options:
A. `13`
B. `13.0`
C. `28`
D. `28.0`
E. Evaluating this expression yields an error

Question 2) [2 marks]

```
'IT' + '5001' * 2
```

Options:
A. `'IT10002'`
B. `'IT50015001'`
C. `'IT5001IT5001'`
D. `None`
E. Evaluating this expression yields an error

Question 3) [2 marks]

```
int('-12.210')
```

Options:
A. `-12.21`
B. `-12`
C. `-13`
D. `None`
E. Evaluating this expression yields an error

Question 4) [2 marks]

```
True and True or False
```

Options:
A. `True`
B. `False`
C. `None`
D. Evaluating this expression yields an error
E. None of the above

Question 5) [2 marks]

```
['a', 'b', 'c', 'd'][::-1]
```

Options:
A. `['d']`
B. `['a', 'b', 'c']`
C. `['d', 'c', 'b', 'a']`
D. `None`
E. Evaluating this expression yields an error

Question 6) [2 marks]

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

Options:
A. `(1, 2, 3, 4)`
B. `((4, 6))`
C. `((8, 9),)`
D. `None`
E. Evaluating this expression yields an error

Question 7) [2 marks]

```
[].append('IT5001')
```

Options:
A. `['IT5001']`
B. `['IT', 5001]`
C. `['I', 'T', '5', '0', '0', '1']`
D. `None`
E. Evaluating this expression yields an error

Question 8) [2 marks]

```
([] + [1] * 2)[:1]
```

Options:
A. `[1]`
B. `[2]`
C. `[[1, 1]]`
D. `None`
E. Evaluating this expression yields an error

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

Question 9) [2 marks] `{0: 1, 2: 3}[0]` Options: A. `0` B. `1` C. `(0, 1)` D. `None` E. Evaluating this expression yields an error	**Question 10)** [2 marks] `3 in {1, 2, {3, 4}}` Options: A. `0` B. `1` C. `True` D. `False` E. Evaluating this expression yields an error
Question 11) [2 marks] `(lambda x: x[-1])('abc')` Options: A. `'a'` B. `'b'` C. `'c'` D. `None` E. Evaluating this expression yields an error	**Question 12)** [2 marks] `(lambda x: x(3))(lambda x: x * 4)` Options: A. `12` B. `x` C. `<function <lambda> at 0x0000...>` D. `None` E. Evaluating this expression yields an error

True or False Questions [16 marks]

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

In each of these questions, you are given either a statement or a Python expression. For each of these, determine if the statement is true or false, and shade the correct option in the 'ANSWER BOOKLET' with a **pencil (at least 2B)**.

Question 13) [2 marks]. The expression `0 < 2 > 1` evaluates to `True`.

Question 14) [2 marks]. The expression `'a dog' >= 'a dog!'` evaluates to `True`.

Question 15) [2 marks]. When iterating over a tuple like `('abc', 'de')` using a single `for` loop, the loop would run for five iterations.

Question 16) [2 marks]. The expression `(i + 1 for i in range(3))` produces the tuple `(1, 2, 3)`.

Question 17) [2 marks]. Dictionaries cannot be added to sets.

Question 18) [2 marks]. Alice usually gives descriptive names to variables, for example, to call a variable `current_user` instead of `cu`. Alice is **wrong** to do so because programs only need to be executable by the computer, and writing longer variable names takes up more space.

Question 19) [2 marks]. Testing your code is important even if you believe that your code works.

Question 20) [2 marks]. Lambda expressions **must** receive at least one positional argument, therefore `lambda: x` is not a syntactically valid expression in Python.

Program Tracing [25 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 21) [8 marks]

```
def num_mults(x, y):
    res = 0
    for i in range(x + 1):
        if not i % y:
            res += 1
    return res
print(num_mults(12, 3))
```

Options:
A. 3
B. 4
C. 5
D. The program does not terminate
E. None of the above

Question 22) [9 marks]

```
def zero_one(ls):
    if len(ls) <= 1: return ls
    if ls[0] and not ls[-1]:
        return (ls[-1:] + zero_one(ls[1:-1])
            + ls[:1])
    if ls[0]:
        return zero_one(ls[:-1]) + ls[-1:]
    return ls[:1] + zero_one(ls[1:])
print(zero_one([1, 1, 0, 1, 0]))
```

Options:
A. `[1, 1, 1, 0, 0]`
B. `[0, 1, 0, 1, 1]`
C. `[1, 0, 1, 1, 0]`
D. `[0, 0, 1, 1, 1]`
E. None of the above

Question 23) [8 marks]

```
def combine(d1, d2):
    res = {}
    for k, v in d1.items():
        res[k] = v
    for k, v in d2.items():
        res[k] = v
    return res
print(combine({1: 2, 2: 3, 3: 4},
              {1: 3, 5: 7}))
```

Options:
A. `{1: 3, 2: 3, 3: 4, 5: 7}`
B. `{1: 2, 2: 3, 3: 4, 1: 3, 5: 7}`
C. `{1: 4, 5: 7}`
D. The program does not terminate
E. None of the above

Program Comprehension [20 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 and write your answers in the 'ANSWER BOOKLET'. You can only obtain full marks for a question if you answer accurately and concisely as well as write legibly.

Observe the following program fragments.

```
1  def f24(a, b):
2      if a < 0 and b < 0:
3          return -a - b
4      if a < 0:
5          return b - a
6      if b < 0:
7          return a - b
8      return a + b
```

```
1  def f25(a, b):
2      res = set()
3      for e in a:
4          res.add(e)
5      for e in b:
6          res.discard(e)
7      return res
```

Question 24 [10 marks]. Assuming the arguments to `f24` are both integers, **describe function `f24`; or in other words, what does `f24` do?**

Question 25) [10 marks]. Assuming the arguments to `f25` are strings, ranges, tuples, lists, sets or dictionaries, **describe function `f25`; or in other words, what does `f25` do?**

Programming [15 marks]

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

Question 26) [15 marks]. A guitar consists of (among other things) six *guitar strings* and a fretboard that consists of *frets*.

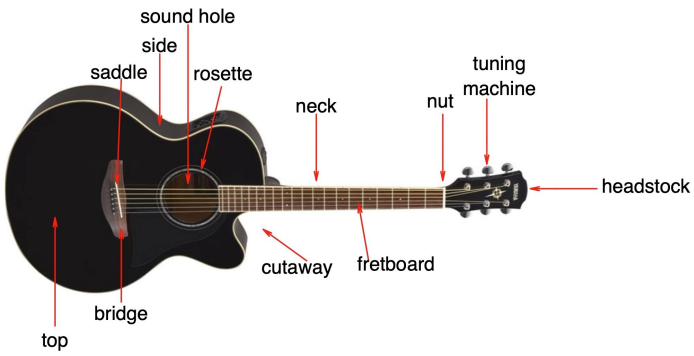
There are (typically) 12 musical notes: A, A#, B, C, etc. We represent this as a list of Python strings `notes` which, in this question, you are assumed to have access to.

```
notes = ['A', 'A#', 'B', 'C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#']
```

Usually, the six *guitar strings* are *tuned* to some *tuning*, which in this question we allow to be a list of Python strings, all of whom are valid notes; for example, from the lowest to highest *guitar string*, they might be tuned to the notes `['E', 'A', 'D', 'G', 'B', 'E']`. Taking for example the lowest (leftmost) *guitar*

string which is tuned to `'E'`, if the *guitar string* is not pressed against any *fret* on the fretboard (also known as fret 0), playing that *guitar string* produces the note `'E'`.

If we were to press that *guitar string* against fret 1, then the note produced when it is played is 1 *semitone* up from `'E'` (which is 1 element to the right of `'E'` in `notes`), so the note produced is `'F'`. In general, when a *guitar string* is tuned to x and it is pressed against fret n, the note produced is n semitones up from x. Also note that one semitone up from `'G#'` is `'A'`.



A guitar. Image source: Yamaha

The `press_strings` function receives 1) `tuning`: a list of six Python strings, all of whom are valid notes, and 2) `fret_presses`: a list of six nonnegative integers. As output, the function returns a list of six elements where the i^{th} element is the note produced when the *guitar string* tuned to `tuning[i]` is played when pressed against the `fret_presses[i]`th fret. Example runs follow:

```
>>> press_strings(['E', 'A', 'D', 'G', 'B', 'E'], [2, 4, 4, 2, 2, 2])
['F#', 'C#', 'F#', 'A', 'C#', 'F#']
>>> press_strings(['A', 'B', 'C', 'D', 'E', 'F'], [2, 1, 2, 2, 1, 2])
['B', 'C', 'D', 'E', 'F', 'G']
```

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

```
1 notes = ['A', 'A#', 'B', 'C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#']
2 def press_strings(tuning, fret_presses):
3     res = <BLANK_1>
4     for i in <BLANK_2>:
5         tune = tuning[i]
6         fret = fret_presses[i]
7         new_note = <BLANK_3>
8         res.append(new_note)
9     return res
```

– End of Assessment –

NATIONAL UNIVERSITY OF SINGAPORE

Department of Computer Science, School of Computing

IT5001—*Software Development Fundamentals*

Academic Year 2022/2023, Semester 2

Mid-Term Assessment

SOLUTIONS MANUAL

11 February 2023 **Time allowed:** 1 hour

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**	●	○	○	○	○

	True	False		True	False		True	False		True	False
13	●	○	**14**	○	●	**15**	○	●	**16**	○	●
17	●	○	**18**	○	●	**19**	●	○	**20**	○	●

	A	B	C	D	E		A	B	C	D	E		A	B	C	D	E
21	○	○	●	○	○	**22**	○	○	○	●	○	**23**	●	○	○	○	○

Program Comprehension

Question 24) [10 marks]. Describe function `f24`; or in other words, what does `f24` do?

Answer: $f24(a,b) = |a| + |b|$.

Question 25) [10 marks]. Describe function `f25`; or in other words, what does `f25` do?

Answer: `f25(a, b)` returns `set(a) - set(b)`.

Programming

Question 26) [15 marks]. Replace each blank with a valid Python expression/statement and write your answers in the table provided below.

```
notes = ['A', 'A#', 'B', 'C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#']
def press_strings(tuning, fret_presses):
    res = []
    for i in range(len(tuning)):
        tune = tuning[i]
        fret = fret_presses[i]
        new_note = notes[(notes.index(tune) + fret) % len(notes)]
        res.append(new_note)
    return res
```

– End of Solutions Manual –