CSI2100-01 Spring 2020 Lab 7m

Due to the Covid19 precautions, this lab (Lab 7m) is taken as a replacement for the midterm exam. The score that you will achieve for this lab will consitute your midterm exam score.

Because of its exam character, Lab 7m differs from our regular labs in the following ways:

- To give equal opportunity to all course participants, no individual counseling is provided for Lab 7m. Please ask all questions on the Q&A board, for everybody to participate and benefit.
- We only provide a subset of all possible test cases with Hyeongjae Python for Lab 7m.
  - You are asked to come up with your own test cases to make sure that your program meets all criteria of the asked solution.
- The number of trial runs on Hyeongjae Python may affect your score. Please refer to Section 1.1 for details.

1 Marking Criteria and Due Date

- **Due date: Tuesday, May 26, 2020, 23:00.**
  This is a *firm* due date. Please submit the deliverables stated in Section 3 on page 6 by this due date on YSCEC.
- We will test your submitted lab solutions with various input data (tests). Score is only given to programs that compile and produce the correct output with Python version 3.
- The score of a lab problem is determined by the number of passed tests.
- Points are deducted for programs that are named wrongly. See the list of deliverables for the required file names.
- Points are deducted for programs that produce warnings.
- Points deductions on programming style: Please provide comments in your code. Use docstrings to comment all functions that you introduce (but it is not mandatory to introduce functions — this is entirely up to you).

- Please pay particular attention to the requested output format of your programs. Deviating from the requested output format results in points deductions.
- This is an individual assignment. All submissions are checked for plagiarism. Once detected, measures will be taken for **all** students involved in the plagiarism incident (including the "source" of the plagiarized code).

1.1 Hyeongjae Python

You can test all problems of this lab on our Hyeongjae Python website. Using Hyeongjae Python is recommended but not mandatory. Using Hyeongjae Python is governed by the following rules.

(**R1**): For each programming problem, you have three *free* trials.

(**R2**): For each trial after the third one, 5% of the problem's score will be deducted from your achieved score for the corresponding lab problem (achieving the solution with fewer trials gives a higher score).

(**R3**): Deductions are capped at a maximum of 50% of a problem's total score.

(**R4**): No deductions apply for the lower 50% of a problem's score.

Please refer to the following table on how deductions are computed.

Achieved score (%)	Number of Trials (×)	Deduction (%)	Score after deduction (%)	Remarks
100	3×	0	100	R1
100	4×	5	95	R2
80	4×	5	75	R2
100	13×	50	50	R2
100	14×	50	50	R2, R3
60	7×	10	50	R2, R4

2 Programming Problems

Problem 1 (25 pts):

A communications technology company charges users for its video conferencing solution an hourly rate. Write a program that computes a person's video conferencing fee according to the following rates (KRW stands for "Korean Won").

Hours x	**Price per hour in KRW**
$0 \leq x < 20$	870
$20 \leq x < 35$	700
$35 \leq x < 100$	400
$x \geq 100$	250

The minimum charge is 3,000 KRW (if a person's accrued expenses are below 3,000 KRW, that person will be charged 3,000 KRW). If the number of participants is larger than four, a 20% surcharge is added. Please output the computed fee with two digits after the fractional point, and please use a comma (',') to group digits-of-three with the whole-number part. The user input for the accrued hours is of type `float` and ≥ 0. The user input for the number of participants is of type `int` and ≥ 1. You do not have to consider faulty user input.

For example:

```
Please enter the number of hours of video conferencing: 10
Please enter the number of participants: 2
Your conferencing fee: 8,700.00 KRW
```

Another example:

```
Please enter the number of hours of video conferencing: 10.0
Please enter the number of participants: 5
Your conferencing fee: 10,440.00 KRW
```

Problem 2 (30 pts):

Consider the following algorithm to generate a sequence of integers. Start with an integer $n > 0$. If n is even, divide it by 2. If n is odd, multiply by 3 and add 1. Repeat with the new value of n until $n = 1$.

For example, for $n = 13$ the following sequence will be generated:

$$13 \quad 40 \quad 20 \quad 10 \quad 5 \quad 16 \quad 8 \quad 4 \quad 2 \quad 1$$

It has been shown that this algorithm will terminate for every integer between 1 and $1,000,000$.

For an input n, the *cycle-length* of n is the length of the generated sequence of numbers up to and including 1. In the above example, the cycle-length of 13 is 10.

Given any two integers i and j, $0 < i \leq j \leq 1,000,000$, compute the maximum cycle-length over all numbers between i and j, including both endpoints.

For example:

```
i, j:  1 25 
Maximum cycle-length: 24
```

As shown in the above example, your program must read the integers i and j from a single line of input, with an arbitrary amount of leading blanks, one or more blanks between i and j, and an arbitrary amount of trailing blanks. No other faults will occur in the input.

Problem 3 (45 pts):

Please consider the below Python code-snippet, which contains the bus departure time-table of a fictional Songdo $\Rightarrow$ Sinchon shuttle-bus service:

```
 1  times = ((36, 48, 56),            # 5:00 -  5:59 (am)
 2           (2, 15, 31, 43),         # 6:00 -  6:59
 3           (2, 17, 34, 46),         # 7:00 -  7:59
 4           (2, 16, 32, 42),         # 8:00 -  8:59
 5           (2, 10, 20, 30, 40, 50), # 9:00 -  9:59
 6           (3, 11, 21, 32, 42, 53), #10:00 - 10:59
 7           (2, 10, 20, 30, 40, 50), #11:00 - 11:59
 8           (2, 16, 32, 42),         #12:00 - 12:59 (pm)
 9           (2, 10, 20, 30, 40, 50), #13:00 - 13:59
10           (2, 12, 22, 32, 42, 52), #14:00 - 14:59
11           (4, 15, 30, 45),         #15:00 - 15:59
12           (3, 13, 32, 47),         #16:00 - 16:59
13           (2, 19, 35, 45),         #17:00 - 17:59
14           (2, 15, 30, 45),         #18:00 - 18:59
15           (2, 30)                  #19:00 - 19:59
16          )
```

Listing 1: Time-table code-snippet

In this time-table, the tuple in line 1 defines that the first bus leaves at `5:36`, followed by buses at `5:48` and `5:56`.

The tuple in line 2 defines that buses leave at `6:02`, `6:15`, `6:31` and `6:43`.

Each tuple (lines 1 ... 15) defines the bus departure times (in minutes) for one particular hour of the day. The last buses of the day leave at the 19th hour, at `19:02` and `19:30`. There are no buses before `5:00 am` and after `19:30`.

Your task is to write a program that asks the user for the preferred departure time and provides the departure-time of the next bus. For example:

```
Enter preferred departure time (hh:mm): 5:00
The next bus leaves at 5:36
```

Another example:

```
Enter preferred departure time (hh:mm): 5:42
The next bus leaves at 5:48
```

Note 1 (user input)

(a) The user input will always contain a valid hour (0–23) and a valid minute (0–59). Hour and minute are separated by exactly one colon `':'`. Nothing else occurs in the input (no faulty input).

(b) Both hours and minutes are specified by **up to 2 digits**. A leading zero is permissible but not required. The following examples show all possible ways for the user to input 7 am:

```
7:0
7:00
07:0
07:00
```

(c) **Hint:** Python's string-to-int conversion using `int()` strips leading zeros, for example:

`0 == int('00')` is `True`
`7 == int('07')` is `True`

Note 2 Your program shall reject all user input before the 5th hour and after the 19th hour of the day. For example:

```
Enter preferred departure time (hh:mm): 4:59
Outside of operation hours!
```

Another example:

```
Enter preferred departure time (hh:mm): 20:00
Outside of operation hours!
```

Note 3 If a user's input is exactly the same as a bus departure time, you shall defer the user to the next bus:

```
Enter preferred departure time (hh:mm): 17:02
The next bus leaves at 17:19
```

Note 4 If there's no more bus in the same hour, the first bus in the next hour must be selected:

```
Enter preferred departure time (hh:mm): 18:45
The next bus leaves at 19:02
```

Note 5 If there's no more bus in the 19th hour, the buses are already outside of the operating hours:

```
Enter preferred departure time (hh:mm): 19:35
Outside of operation hours!
```

Note 6 With your program's **output**, **minutes** must always be displayed in **2-digit format**. Single-digit hours (0–9) must be displayed with one digit:

```
Enter preferred departure time (hh:mm): 7:00
The next bus leaves at 7:02
```

Note 7 (code skeleton)
For this problem, you are provided with two files: `lab_m_p3.py` and `data_generator.py`. Please download both files from YSCEC and add them to the PyCharm project on your local computer.

- File `lab_m_p3.py` contains a code skeleton that you must extend to solve this problem. The code skeleton is depicted in Listing 2. Please provide your code after line 9.
  - Please note: you are **not allowed to change anything above line 9** in file `lab_m_p3.py`, otherwise automated grading will fail.
  - The file `lab_m_p3.py` is the *main program* that you must run in PyCharm. (Do not attempt to run the data generator in file `data_generator.py`.)
- As shown in Listing 2 (lines 1...4), the bus departure times (tuple `times`) are pre-loaded from the data generator. It is not necessary to understand the details of this code beyond the explanations provided below.
  - Function `getDepartTimes()` is located in file `data_generator.py`. It returns the tuple that contains the departure times.
    - The code of function `getDepartTimes()` is self-explanatory. Please have a look in file `data_generator.py`.

    * In **Note 9** below it is described how you can extend the data generator with different departure times (for testing your code).
  - After the function call in line 4 of Listing 2, tuple “`times`” contains the data depicted in Listing 1.

```
1  from data_generator import getDepartTimes
2  import sys
3
4  times = getDepartTimes()
5
6  inp = input('Enter preferred departure time (hh:mm): ')
7  #
8  # Your code here:
9  #
```

Listing 2: Skeleton code

Note 8 (departure times)
The departure times used for grading may be different from those in Listing 1. You are not allowed to make any assumptions in your code, except the following:

(a) The tuple named “`times`” (see line 4 in Listing 2) will always contain 15 subtuples, representing the hours from 5:00 (am) to 19:00.

(b) Each subtuple will contain at least one bus departure time.

Note 9 (additional test cases)
You are recommended to come up with additional bus departure times to thoroughly test your program.

- Please have a look at the data generator in file `data_generator.py`. This file provides ample comments to guide you where to put additional test cases.
- The value of the global variable “`selection`” in line 3 of file `data_generator.py` selects a particular test case.
- Please refer to the comments inside the if-statement of function `getDepartTimes()` on how to add a test case.

3 Deliverables

Problem	File name	Hyeongjae Python
1	lab7m_p1.py	✓
2	lab7m_p2.py	✓
3	lab7m_p3.py	✓

- Please submit all files in a ZIP-file named lab7m_<student_id>.zip on YSCEC.
- Please refer to Section 1 on page 1 for the lab due date.