Instruction to candidates:

Your program code and output for each of Task 1 to N should be saved in a single `.ipynb` file using Jupyter Notebook. For example, your program code and output for Task 1 should be saved as:

`Task1_<your name>_<centre number>_<index number>.ipynb`

Make sure that each of your `.ipynb` files shows the required output in Jupyter Notebook.

1 Name your Jupyter Notebook as

`Task2_<your name>_<centre number>_<index number>.ipynb`

A physical fitness test for school students consists of six items:

- sit-ups
- standing broad jump
- sit-and-reach
- pull-ups
- shuttle run
- 2.4 km run

The performance for each item is measured on a scoring system, which is age and gender specific. The scoring system consists of five bands, A to E. Each band is demarcated by an **excluded** minimum requirement. Attaining band A will earn a student 5 points and attaining band E will earn a student 1 point. If a student fails to meet the minimum requirement for band E in an item, he or she will not be awarded any point for that item.

For example, the scoring system for a 12 year old male student doing sit-ups is as follows:

Band	Minimum No. of Sit-Ups	Points Awarded
A	41	5
B	35	4
C	31	3
D	26	2
E	21	1

The task is to design a program to process the results of the physical fitness test.

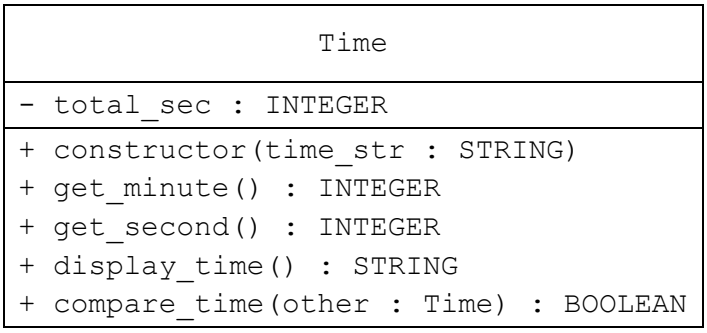
For each of the sub-tasks, add a comment statement at the beginning of the code using the hash symbol '#', to indicate the sub-task the program code belongs to, for example:

```
In [1] :  # Task 2.1
          Program code
```

```
          Output:
```

Task 2.1

Implement the `Time` class based on the following class diagram. The descriptions for some of the methods can be found below:



Methods	**Description**
`display_time() : STRING`	Returns a string in the format of `"mm:ss"`. For example, `"15:03"`, `"02:52"`.
`compare_time(other : time) : BOOLEAN`	Compares the current `Time` object with another `Time` object `other`. Returns `True` if current `Time` object represents a longer timing as compared to `other`, `False` otherwise.

[4]

Task 2.2

Implement the class `BandingList` which has the following attributes and methods.

Identifier	Data Type	Descriptions
`test_item`	`STRING`	A string to describe the assessment item, such as `"Shuttle Run Time"`.
`op`	`STRING`	A string to store the operator for comparison of this item. For example, for standing broad jump, the longer the distance, the better. The operator stored will be `">"`. On the contrary, for running, the shorter the timing the better. Then a `"<"` operator will be stored.
`boundary_values`	`LIST`	A list of boundary values, with the first one as the **excluded** minimal requirement for band A, and last one as the **exclusded** minimal requirement for band E.

Methods	Description

`test_band() : STRING`	Returns a string displaying the test item followed by a banding list.
	
	The name of each test item should occupy the first 20-character spaces.
	
	This shall be followed by the banding list with each value of the list occupying 5-character spaces, from band A to band E.
	
	Each value is then separated from the next using the `"|"` character
	
	For example:
	`Sit-ups: 42 |37 |33 |28 |24 |`
	`2.4KM Run: 11:31|12:31|13:41|14:51|16:01|`

`calc_point (record) : INTEGER`	Takes in a record value and compare against the banding list. Determines which band the record belongs to. Returns the corresponding number of points awarded

[4]

Write additional code to implement a function `trial_banding_list()` that:

- creates a `BandingList` object
- displays the test item for which the `BandingList` object is created for, and the corresponding banding list.

Write further code implement three test cases to check whether the `calc_point()` method has been correctly implemented. Justify the rationale for each test case using comments.

[5]

Task 2.3

Implement the class `TestStandard` which has the following attributes and methods. You may implement additional mutator and accessor methods where deemed appropriate.

Identifier	Data Type	Descriptions
`age`	`INTEGER`	Age group
`gender`	`STRING`	`"M"` for male and `"F"` for female
`bd_lists`	`LIST`	A list of `BandingList` objects. Each `BandingList` object corresponds to one of the assessment item that is appropriate for the `gender` and `age`.

Methods	Description

`test_std() : STRING`	Returns a string containing 7 lines.
	
	The first line should indicate the age group and gender for which the test standard applies.
	
	The following 6 lines should display the banding list for each assessment item.
	
	The name of each test item should occupy the first 20-character spaces.
	
	This shall be followed by the banding list with each value of the list occupying 5-character spaces, from band A to band E.
	
	Each value is then separated from the next using the `"|"` character
	
	For example:
	`Age: 12, Gender: M`
	`Sit-ups: 41 |35 |31 |26 |21 |`
	`Standing Broad Jump:202 |188 |175 |162 |149 |`
	`Sit-and-Reach: 39 |35 |31 |27 |22 |`
	`Pull-ups: 24 |20 |15 |10 |4 |`
	`Shuttle Run: 10.4 |11.0 |11.4 |11.8 |12.3 |`
	`2.4KM Run: 12:01|13:11|14:21|15:31|16:51|`

[4]

The files `"test_standard_male.txt"` and `"test_standard_female.txt"` contains information on the test standards by age and item for male and female students respectively. The tabular view of the data in the files is shown below:

age	band	point	Sit-ups	Standing Broad Jump	Sit and Reach	Pull-ups	Shuttle Run	2.4KM Run
operator	nil	nil	>	>	>	>	<	<
12	A	5	41	202	39	24	10.4	12:01
	B	4	35	188	35	20	11	13:11
	C	3	31	175	31	15	11.4	14:21
	D	2	26	162	27	10	11.8	15:31
	E	1	21	149	22	4	12.3	16:51
13	A	5	42	214	41	25	10.3	11:31
...	...	...	...	...	...	...	...	...

You may open the two files to study the actual format of the data.

Write additional code to implement a function `init_standard(file_name)` that:

- takes in a file `file_name`, containing data stored in a format identical to that of `"test_standard_male.txt"` and `"test_standard_female.txt"`
- processes and stores the data for each age group and gender as a `TestStandard` object
- returns a list of all the `TestStandard` objects created.

[6]

Write further code to implement a function `trial_init_standard()` to:

- test your `init_standard()` function and `TestStandard` class by:
  - processing `"test_standard_male.txt"` and `"test_standard_female.txt"`
  - displaying all the processed data using the `test_std()` method of the `TestStandard` class.

[3]

Task 2.4

Implement the class `StudentRecord` which has the following attributes and methods. You may implement additional attributes, and mutator and accessor methods where deemed appropriate.

Identifier	Data Type	Descriptions
`name`	`STRING`	Name of student
`age`	`INTEGER`	Age of student
`gender`	`STRING`	`"M"` for male and `"F"` for female
`test_results`	`LIST`	A list containing the individual test results of the 6 items of the student

<table>
<tr><th>Methods</th><th>Description</th></tr>
<tr><td><code>process_score() : STRING</code></td><td>Based on the student's age and gender, compare his/her performance in each item against the corresponding test standards.

Calculate the award attained based on the following standards:

<table>
<tr><th>Award</th><th>Minimum Criteria</th></tr>
<tr><td>Gold</td><td>At least band C in each of the six items
Achieved a minimum total of 21 points</td></tr>
<tr><td>Silver</td><td>At least band D in each of the six items
Achieved a minimum total of 15 points</td></tr>
<tr><td>Bronze</td><td>At least band E in each of the six items
Achieved a minimum total of 6 points</td></tr>
</table>

No award is attained if the student does not meet the minimum criteria for the Bronze award.</td></tr>
<tr><td><code>results() : STRING</code></td><td>Returns a string containing

• the student's name, gender, age and award type in the first line,
• the test performance for each item in the second line,
• the corresponding points attained for each item in the third line

For the second and third lines, each value will occupy 5-character spaces. Each value is then separated from the next using the <code>"|"</code> character.

For example:
<pre>Maira Creager, F, 12: Gold
37 |207 |57 |13 |10.9 |13:27|
5 |188 |5 |4 |5 |5 |</pre></td></tr>
</table>

【7】

The file "student_test_records.txt" contains the physical fitness test results of fifty students. The first line in the file contains the headers for each record that is stored in the following order:

```
name,gender,age,Sit-ups,Standing Broad Jump,Sit and Reach,Pull-ups,Shuttle Run,2.4KM Run
```

Each subsequent line in the file contains the test record for one student.

Write additional code to implement a function process_student_record() that:

- reads student data from the file "student_test_records.txt",
- processes each record as a StudentRecord object
- write each record using the results() method of the StudentRecord class into another file "student_result.txt".

After processing all the records, write additional code to tally the results into the following table format at the end of "student_result.txt", using a cell width of 10 character spaces. Each cell is then demarcated from the adjacent cell using the "|" character.

```
          |No Award  |Bronze    |Silver    |Gold
Male      |S         |T         |U         |V
Female    |W         |X         |Y         |Z
```

[7]

Save your Jupyter Notebook for Task 2.