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

`Task3_<your name>_<centre number>_<index number>.ipynb`

Bowling is a sport in which a 'bowler' rolls a bowling ball down a synthetic lane and towards ten pins positioned at the end of the lane. The objective is to score points by knocking down as many pins as possible.

A bowling game
One game of bowling consists of ten frames. Each frame consists of two chances for the bowler to knock down ten pins.

Strikes and spares
Knocking down all ten pins on the first roll of any frame is called a **strike**, denoted by 'X' on the score sheet. If a bowler takes two rolls to knock down all ten pins, it is called a **spare**.

Scoring and bonus scoring
Each pin that is knocked down is worth 1 point.
A strike is worth 10 points plus the number of pins hit on the next two rolls.
A spare is worth 10 points plus the number of pins hit on the next roll.
The total score for a game ranges from 0 to 300 points.

The tenth frame
A bowler who strikes or spares the tenth frame will be given one extra roll. The number of pins hit on this roll will be added to the bowler's score.

Sample Scores

Frame	1	2	3	4	5	6	7	8	9	10
Result	**X**	7|**3**	7|2	9|**1**	**X**	**X**	**X**	2|3	6|**4**	7|**3**|3
Frame Score	20	17	9	20	30	22	15	5	17	13
Cumulative Score	20	37	46	66	96	118	133	138	155	168

Frame	1	2	3	4	5	6	7	8	9	10
Result	0|5	8|0	**X**	0|5	**X**	6|**4**	0|5	8|1	9|**1**	5|0
Frame Score	5	8	15	5	20	10	5	9	15	5
Cumulative Score	5	13	28	33	53	63	68	77	92	97

Frame	1	2	3	4	5	6	7	8	9	10
Result	**X**	**X**	**X**	**X**	9|**1**	**X**	0|0	2|2	8|**2**	**X**|X|X
Frame Score	30	30	29	20	20	10	0	4	20	30
Cumulative Score	30	60	89	109	129	139	139	143	163	193

The following algorithms calculate the total score for a bowling game.

```
//Used interchangeably in the code for readability
Ten ← 'X'
Strike ← Ten

//Converting the 'X' or 'number' to INTEGER
FUNCTION Pins(Throw as STRING)
  IF Throw = Ten THEN
    RETURN 10
  ELSE
    RETURN INTEGER(Throw)
  ENDIF
END FUNCTION

//Recursive Procedure
FUNCTION Bowling_Score(Throws as STRING)

  //Helper function to keep track of current frame number
  FUNCTION Bowling_Score_Helper(Throws as STRING, Frame_Num as INTEGER)

    //Frame 10 with no bonus
    IF Frame_Num = 10 AND LENGTH(Throws) = 2 THEN
      RETURN SUM(Pins(Throws[0]), Pins(Throws[1]))
    ENDIF

    //Frame 10 with bonus
    IF Frame_Num = 10 AND LENGTH(Throws) = 3 THEN
      RETURN SUM(Pins(Throws[0]),Pins(Throws[1]),Pins(Throws[2]))
    ENDIF

    //A Strike
    IF Throws[0] = Strike THEN
      Frame_Score ← 10 + SUM(Pins(Throws[1]), Pins(Throws[2]))
      RETURN Frame_Score + Bowling_Score_Helper(Throws[1:], Frame_Num + 1)
    ENDIF

    Frame_Score ← SUM(Pins(Throws[0]), Pins(Throws[1]))

    //A Spare
    IF Frame_Score = 10 THEN
      RETURN 10 + Pins(Throws[2])+ Bowling_Score_Helper(Throws[2:], Frame_Num + 1)
    ENDIF

    //Frame with no bonus
    RETURN Frame_Score + Bowling_Score_Helper(Throws[2:], Frame_Num + 1)

  END FUNCTION

  RETURN Bowling_Score_Helper(Throws, 1)

END FUNCTION
```

Note: The above pseudocode is available in the text file `PSEUDOCODE_TASK_3_1.TXT` but with `'='` used in place of `'←'` shown above.

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 3.1
            Program code

            Output:
```

Task 3.1

Write a program to calculate a bowling score using the algorithms provided on the previous page.

Test your program with the following calls:

- `Bowling_Score('X2815X91X365452X0X')`
- `Bowling_Score('91739182X90X90X82X')`

[7]

Task 3.2

Make any necessary amendments to your code in **Task 3.1** to ensure that it can handle all possible scenarios. Include comments on why each amendment was made.

Further test your program with three suitable test cases to ensure that it is able to handle all possible scenarios.

[9]

At the 2018 Singapore Championship, bowlers play a total of six games each in the qualifying round. The eight bowlers with the highest total score qualify for the Masters Competition. You may assume that there are no bowlers with the same total score.

`SCORES.TXT` is a text file containing the register number, country and the scores for six games of twenty bowlers in the qualifying round, with one bowler per line in the format:

`<Register Number> <Country> <Score 1> <Score 2> … <Score 6>`

For example:

`157 MAS XXXXX9091XXX82 90XXXXXXXXXX9 … 8172XXXX919191XX8`

Task 3.3

Design and write program code to:

- read the entire contents of `SCORES.TXT`
- calculate the score of each game for all twenty bowlers, using your code in **Task 3.1**, assuming that all the scores in `SCORES.TXT` are valid scores
- calculate the total score for each bowler and store this in an appropriate data structure together with the respective `Register Number` and `Country`
- use **bubble sort** to sort the bowlers according to their total score
- output the Official Results of the competition.

An example of the output should look like this:

```
Official Results:
```

Position		Register Number		Country		Total Score
1		175		SIN		1734
2		299		SIN		1724
...		...		...		...
20		245		MAS		1270

[11]

Save your Jupyter Notebook for Task 3.