Pre-WA1 Practice Paper 3 (50 marks, recommended duration – 1 hour 15 minutes)

Answer **all** questions.

All tasks must be done in the computer laboratory. You are not allowed to bring in or take out any pieces of work or materials on paper or electronic media or in any other form.

Approved calculators are allowed.

Save each task as it is completed.

The use of built-in functions, where appropriate, is allowed for this paper unless stated otherwise.

The number of marks is given in brackets [ ] at the end of each question or part question.

Your program code and output for each of Task 1 to 3 should be saved in a single .ipynb file. For example, your program code and output for Task 1 should be saved as

`TASK1_<your name>_<centre number>_<index number>.ipynb`

1 Name your Jupyter Notebook as

`TASK1_<your name>_<centre number>_<index number>.ipynb`

The text file `student.txt` contains information of some students in the following format.

```
Name,Age,Gender
```

The task is to process the data in the file.

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 1.1
         Program code

         Output:
```

Task 1.1

Write program code to read in the contents of the file `students.txt` and organise the data into an appropriate nested list. [3]

Task 1.2

Write program code to:

- prompt the user to input the name of a student
- display the information of the specified student in the following format:
  `Age: 16 Gender: M`
- or display a suitable error message if the student cannot be found and prompt the user for the name of another student. [4]

Task 1.3

Write program code to:

- calculate the average age of the students in the file and display it to 1 decimal place
- calculate the number of males and females in the class and display it. [6]

Task 1.4

Write program code to:

- capitalise the first letter of every name
- create a new file `updated_students.csv` to output the data in the following format:

  `Capitalized_Name,Age,Gender` [4]

Save your Jupyter notebook for Task 1.

2 Name your Jupyter Notebook as

`TASK2_<your name>_<centre number>_<index number>.ipynb`

The task is to create a guess-the-word game.

The program starts by stating the number of letters the word has. At each turn, the player can guess a letter or guess the word. If the player enters a letter, the program displays whether the letter exists or not in the word.

If the letter exists, the program reveals all occurrences of the letter in the word.

If it does not exist or the player has already entered the same letter before, the program displays an appropriate message for each of these two scenarios.

If the player guesses the word wrongly, the program displays an appropriate message.

If the player guesses correctly, the program displays an appropriate message, state the number of guesses the player has made and exits.

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

Write a function `generate_word()` that returns a random word from a list of words, in uppercase. Use this list of words: `[apple, pear, banana, Minion, OMG].` [2]

Task 2.2

Write a function `letter_guess(word, letterList, letter)` that:

- accepts three parameters:
  - `word`, a string representing the word to be guessed
  - `letterList`, a list of letters that the player has already guessed over the course of the game
  - `letter`, a string representing the player's new letter guess.
- checks whether the `letter` exists in the `word`:
  - if the `letter` exists, updates the `letterList` with the new `letter` and updates the `guessedWord` which is a string that reveals to the player all the letters of the word that the player has guessed correctly. Unguessed letters are displayed with underscore
  - if the `letter` appears more than once, update the `letterList` with the new `letter` and updates the `guessedWord` accordingly, i.e. reveals all the occurrences of the `letter`
  - if it does not exist in the `word` at all, display the message "`Sorry. The word does not contain the letter '<insert letter> '`".
- returns the updated `guessedWord`. [6]

Task 2.3

Write program code that:

- calls the `generate_word()` function
- displays the number of letters the generated word contains
- asks the player to enter a letter or guess the word.
- if the letter or word has been entered already, display the message "`You already guessed '<insert letter or word>'`".
- if a letter is entered,
  - call the `letter_guess` function
  - return and display the output of the `letter_guess` function
  - and the output of the `letter_guess` function matches the word, display the message "`Correct! The word is '<insert word>'. Number of tries: <insert number of tries>`" and exit the program.
- if the player inputs a word,
  - and it matches the generated word, display the message "`Correct! The word is '<insert word>'. Number of tries: <insert number of tries>`" and exit the program.
  - but the word is guessed wrongly, display the message "`Incorrect! Try again`". [7]

Save your Jupyter notebook for Task 2.

3 Name your Jupyter notebook as

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

A text file, `TIDES.TXT`, contains the low and high tide information for a coastal location for each day of a month. Each line contains tab-delimited data that shows the date, the time, whether the tide is high or low and the tide height in metres.

Each line is in the format:

`YYYY-MM-DD\tHH:mm\tTIDE\tHEIGHT\n`

- The date is in the form YYYY-MM-DD, for example, 2019-08-03 is 3rd August, 2019
- The time is in the form HH:mm, for example, 13:47
- TIDE is either HIGH or LOW
- HEIGHT is a positive number shown to one decimal place
- `\t` represents the tab character
- `\n` represents the newline character

The text file is stored in ascending order of date and time.

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 program code to:
- read the tide data from a text file
- find the highest high tide and print this value
- find the lowest low tide and print this value.

Use `TIDES.txt` to test your program code. [9]

Task 3.2

The tidal range is the difference between the heights of successive tides; from a high tide to the following low tide or from a low tide to the following high tide.

Copy and amend your program code in **Task 3.1** to:

- output the largest tidal range and the date on which the second tide occurs
- output the smallest tidal range and the date on which the second tide occurs.

Use `TIDES.txt` to test your program code. [9]

Save your Jupyter notebook for Task 3.