BOON LAY SECONDARY SCHOOL

PRELIMINARY EXAMINATION

2022

Name	
CCA	

Subject	:	**COMPUTING**
Paper No	:	**2 (LAB-BASED)**
Subject Code	:	**7155/02**
Level	:	**SECONDARY FOUR EXPRESS**
Date/Day	:	**26 AUGUST 2022 / FRIDAY**
Time	:	**0945 – 1215**
Duration	:	**2 HOURS 30 MINUTES**

Additional Materials:	Electronic version of T1_MOBILE_PLANS.xlsx
	Electronic version of T2_BALANCER.py
	Electronic version of T3_PARITY_INDEXNO_NAME.py
	Electronic version of Quick Reference Glossary

READ THESE INSTRUCTIONS FIRST

Before you start your exam, check that you have received the correct paper and the number of printed pages are correct.

Write your name, index number, and CCA in the spaces at the top of this page.

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.

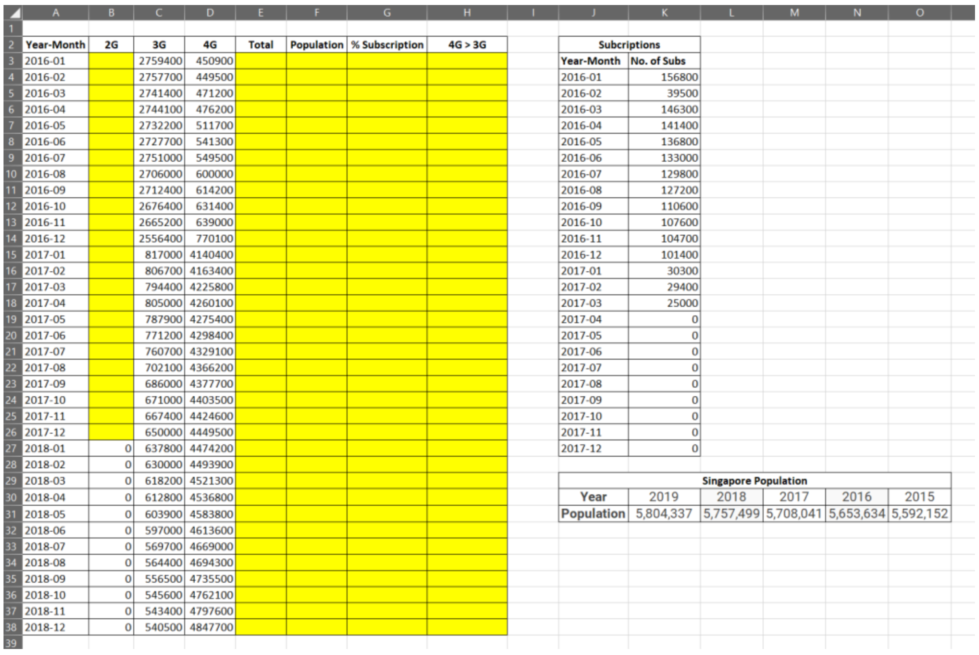
Programs are to be written in Python.
Save your work using the file name given in the question as and when necessary.

The number of marks is given in brackets [ ] at the end of each question or part question.
The total number of marks for this paper is 50.

This document consists of **6** printed pages.

Task 1

Table A2:G38 shows the number of people in Singapore with subscriptions for 2G, 3G and 4G mobile plans between 2016 and 2018.



	A	B	C	D	E	F	G	H
1								
2	Year-Month	2G	3G	4G	Total	Population	% Subscription	4G > 3G
3	2016-01		2759400	450900				
4	2016-02		2757700	449500				
5	2016-03		2741400	471200				
6	2016-04		2744100	476200				
7	2016-05		2732200	511700				
8	2016-06		2727700	541300				
9	2016-07		2751000	549500				
10	2016-08		2706000	600000				
11	2016-09		2712400	614200				
12	2016-10		2676400	631400				
13	2016-11		2665200	639000				
14	2016-12		2556400	770100				
15	2017-01		817000	4140400				
16	2017-02		806700	4163400				
17	2017-03		794400	4225800				
18	2017-04		805000	4260100				
19	2017-05		787900	4275400				
20	2017-06		771200	4298400				
21	2017-07		760700	4329100				
22	2017-08		702100	4366200				
23	2017-09		686000	4377700				
24	2017-10		671000	4403500				
25	2017-11		667400	4424600				
26	2017-12		650000	4449500				
27	2018-01	0	637800	4474200				
28	2018-02	0	630000	4493900				
29	2018-03	0	618200	4521300				
30	2018-04	0	612800	4536800				
31	2018-05	0	603900	4583800				
32	2018-06	0	597000	4613600				
33	2018-07	0	569700	4669000				
34	2018-08	0	564400	4694300				
35	2018-09	0	556500	4735500				
36	2018-10	0	545600	4762100				
37	2018-11	0	543400	4797600				
38	2018-12	0	540500	4847700				
39								

Subcriptions	
Year-Month	No. of Subs
2016-01	156800
2016-02	39500
2016-03	146300
2016-04	141400
2016-05	136800
2016-06	133000
2016-07	129800
2016-08	127200
2016-09	110600
2016-10	107600
2016-11	104700
2016-12	101400
2017-01	30300
2017-02	29400
2017-03	25000
2017-04	0
2017-05	0
2017-06	0
2017-07	0
2017-08	0
2017-09	0
2017-10	0
2017-11	0
2017-12	0

Singapore Population					
Year	2019	2018	2017	2016	2015
Population	5,804,337	5,757,499	5,708,041	5,653,634	5,592,152

Open the file **T1_MOBILE_PLANS.xlsx**

1 In cells **B3:B26**, use vlookup() to fill in the number of 2G subscriptions for each respective month from January 2016 to December 2017 by looking up the data in table **J2:J27**. [2]

2 In cells **E3:E38**, sum up the total number of people with 2G, 3G and 4G subscriptions in each month. [1]

3 In cells **F3:F38**, use hlookup() to fill in the population for each corresponding year in the row by looking up the data in **J29:J31**. [2]

4 In cells **G3:G38**, determine the percentage of the total population who own either a 2G, 3G or 4G subscription in that month. Round your answer to 2 decimal places. [3]

5 In cells **H3:H38**, determine if there are more people who own 4G subscriptions than 3G subscriptions. Display a “T” if it is true and “F” if it is false in the month. [2]

Save your file as **T1_MOBILE_PLANS_INDEXNO_NAME.xlsx** and close your file.

Task 2

A program was written to determine if two strings are balanced.

Two strings are considered balanced if all the characters in the second string can be found inside the first string, regardless of the order and number of appearances.

```
while True:
    print("Welcome to the string balancer")
    st1 = input('string 1: ')
    st2 = input('string 2: ')

    check = True
    for char in st2:
        if char not in st1:
            check = False

    if check == True:
        print("Strings are balanced")
    else:
        print("Strings are not balanced")

    if input("again? (y/n): ")!= 'y':

        break
```

Open the file **T2_BALANCER.py**

Modify the program such that

7 it outputs the length of *st1* and *st2* after each of them has been keyed in. [3]

8 it compares the length of both strings and outputs which string is longer. [2]

9 accepts both “y” and “Y” when it prompts the user if they would like to compare another set of strings. [2]

Save your work as **T2_Balancer_01_INDEXNO_NAME.py**.

Save a copy of your file as **T2_Balancer_02_INDEXNO_NAME.py**

10 Modify your program such that it stores *st1* and *st2* in a list instead of two separate variables.

Modify the rest of the program such that it uses the new list at appropriate junctures. [3]

Save your work.

Task 3

11 The code below was written to determine the parity bit for a given 7-bit binary word.

In a system using even parity, the total number of ‘1’s in a 8-bit word must be even.

A 7-bit binary word is entered and the final 8th bit is determined by counting the number of ‘1’s in the original 7 bits.

If the original 7 bits contains an even number of ‘1’s, the parity bit is 0.

If the original 7 bits contains an odd number of ‘1’s, the parity bit is 1.

Open the file **T3_PARITY_INDEXNO_NAME.py**

There are some errors in the code.

Identify and correct the errors so that it functions according to the description.

Notes:

You may assume that the input for *word* is always correct. [10]

Task 4

A program has been commissioned to generate some statistics for a list of numbers that needs to be input by the user.

12 Write a program that fulfils the following specifications:

- Displays an appropriate welcome message for the program
- Asks the user to input some numbers.
  The number can either be input in the same string all at once, or one at a time.
  The sample below shows all the numbers being entered in a single input.
- Output the average of the list of numbers

- You must make use of at least three functions including the following:

Function (noun)	**Function** (verb)
main()	this is the main program
user_input()	Prompts the user to input numbers and returns a list of numbers to main()
average()	Calculates the average of the numbers in the list and returns the average to main()

Save your program as **T4_SIMPLENUM_INDEXNO_NAME.py**
A sample run of the code is shown below:

Sample 1

```
Welcome to Simple Numbers
Numbers please.
Separate each number using spaces: 1 2 3 4 5 6 7 8 9 10
Average: 5.5
```

[10]

Sample 2

```
Welcome to Simple Numbers
Numbers please.
Separate each number using spaces: 10 20 30 40 50
Average: 30.0
```

13 When your program is complete, input the following numbers as tests for your program and take screenshots of the program.

- Test 1
  Input numbers: `1, 2, 3, 4, 5, 6, 7, 8, 9, 10`
  Save this screenshot as:
  T4_TEST1_INDEXNO_NAME

- Test 2
  Input numbers: `10, 20, 30, 40, 50` [2]
  Save this screenshot as:
  T4_TEST2_INDEXNO_NAME

Save your files either in **.png** or **.jpg** format.

Save your program as **T4_SIMPLENUM_INDEXNO_NAME.py**

14 Extend your code to:

- output the maximum and minimum numbers in the list
- implement a check to ensure that the inputs are valid digits before converting them to the necessary data types.

[8]