Name: _____ **Class:** 4S3 **Index Number:** _____

Anglo-Chinese School (Barker Road)

PRELIMINARY EXAMINATION 2020

SECONDARY FOUR EXPRESS

COMPUTING PAPER 2

7155/02

2 HOUR 30 MINUTES

INSTRUCTIONS TO CANDIDATES

Additional Materials: Electronic version of CARORDERS.XLSX data file
Electronic version of BMI.PY Python file
Electronic version of MARKS.PY Python file
Insert Quick Reference Glossary

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 **7** printed pages, inclusive of the cover page*

Task 1

ABC Company uses a spreadsheet to record the car orders and loans from customers. You are required to finish setting up the spreadsheet to calculate the monthly instalment for each of the order.

ABC Company Car Orders for July 2020

Order Number	Model Type	Package	Basic Price	Package Price	Total Price	Loan %	Loan Amount	Loan Tenure (Years)	Rate	Monthly Instalment
S1001	1.5L Hatchback	Classic				50%		5		
S1002	1.5L Sedan	Classic				40%		7		
S1003	1.5L Hatchback	Deluxe				30%		5		
S1004	1.5L Sedan	Luxury				20%		5		
S1005	2.0L 2WD	Deluxe				20%		4		
S1006	2.0L 2WD	Deluxe				50%		3		
S1007	2.0L Sedan	Luxury				40%		2		
S1008	2.5L Wagon	Classic				20%		3		
S1009	2.5L Wagon	Classic				40%		2		
S1010	1.5L Hatchback	Classic				50%		5		
S1011	1.5L Sedan	Classic				50%		4		
S1012	1.5L Sedan	Luxury				40%		7		
S1013	1.5L Sedan	Classic				30%		6		
S1014	1.5L Hatchback	Classic				50%		1		
S1015	2.5L Wagon	Classic				30%		5		
S1016	2.0L Sedan	Deluxe				40%		4		
S1017	2.0L Sedan	Deluxe				20%		3		
S1018	1.5L Sedan	Luxury				20%		5		
S1019	1.5L Hatchback	Deluxe				50%		5		
S1020	2.5L Wagon	Classic				40%		4		

Model Basic Price

Model	1.5L Hatchback	1.5L Sedan	2.0L Sedan	2.0L 2WD	2.5L Wagon
Basic Price	$83,188	$88,188	$110,188	$120,188	$154,388
Numbers sold					

Rates

Loan Period (Years)	Description	Annual Interest Rate
1	One year	2.00%
2	Two or Three years	1.75%
4	Four years	1.50%
5	Five years or longer	1.25%

Open the file **CARORDERS**. You will see the following data.

Save the file as **ORDERS**_<your name>_<index number>

1 Use an appropriate function to search for the **Basic Price** of each car model type in the **Model Basic Price** table and use it to complete the **Basic Price** column. [2]

2 Use a conditional statement to determine the **Package Price** for each order. The **Package Price** costs 0%, 10% and 20% of the **Basic Price** for Classic, Deluxe and Luxury packages respectively. [2]

3 Thereafter, calculate the **Total Price** of the car. The **Total Price** is the **Basic Price** added to the **Package Price**. [1]

4 Use an appropriate function to calculate the **Loan Amount** of the car. The **Loan Amount** is the loan % of the **Total Price**. [1]

5 Use an appropriate function to search for the **Annual Interest Rate** in the **Rates** table and use it to complete the **Rate** column. [2]

6 Use an appropriate function to calculate the monthly instalment amount and use it to complete the **Monthly Instalment** column. [1]

7 Use an appropriate function to find out the **Numbers sold** for each car model in the **Model Basic Price** table. [1]

Save and close your file.

Task 2

The following program allows the weights and heights of 10 students to be input. The program then computes the body mass index, bmi and output the appropriate output if the bmi falls outside the acceptable range.

```
students = 10
upp_bound = 25
low_bound = 18.5

for count in range(students):
    weight = float(input('Enter weight of student in kg '))
    height = float(input('Enter height of student in cm '))
    bmi = weight/height**2 * 10000
    if bmi > upp_bound:
        print('Student is overweight')
    if bmi < low_bound:
        print('Student is underweight')
```

Open the file **BMI.py**

Save the file as **MYBMI**_<your name>_<index number>

8 Edit the program so that it:

(a) Accepts input for 15 students. [1]

(b) Prints out the message `'Student's weight is normal'` when the bmi falls within the acceptable range of 18.5 to 25 inclusive. [2]

(c) Prints out the number of students that were underweight, as well as the number that were overweight after all the weights and heights of the students have been entered. [2]

Save your program.

9 Save your program as **MYBMI2**_<your name>_<index_number>
Edit your program so that it:

(a) Tests whether the user has entered a weight from 30 kg to 150 kg inclusive, and if not, outputs the message `'Invalid weight'` and asks the user for input again as necessary. [2]

(b) Tests whether the user has entered a height from 80 cm to 200 cm inclusive, and if not, outputs the message `'Invalid height'` and asks the user for input again as necessary. [2]

(c) Works for any number of students. [1]

Save your program.

Task 3

The following program searches a list of names to check if it contains a name.

If the name is found in the list, a message is displayed on the screen that states the corresponding mark in a separate mark list. Otherwise, a message is displayed on the screen that states the name is not in the list.

There are several syntax errors and logical errors in the program.

```
nlist = ["Alden", "Belle", "Charles", "Dolly", "Elle", "Falken", "Grace",
"Hacken"]
mlist = (56, 64, 23, 78, 53, 46, 98, 33]

to_find = input("Which name would you like to search for? ")

items = len(to_find)
num = 0
name_found = False
while name_found == True:
    while num > items:
        if nlist[num] = to_find
            print("{} score {} for the test".format(nlist[num], mlist[num -
1]))
            name_found = True
            num = num
        elif num == items - 1:
            print("{} is not in the list".format(to_find))
            name_found = False
            num = items
        else:
            num = items
```

Open the file **MARKS.py**

Save the file as **MYMARKS**_<your name>_<index number>

10 Identify and correct the errors in the program so that it works correctly according to the rules above. [10]

Save your program.

Task 4

You have been asked to create a daily time-in/time-out program for temporary staff.

The program should:

- Enter the name of temporary staff. There is no need to validate the name.
- Enter the time-in and time-out of the temporary staff in the format of HH:MM. An example is 08:05
- Ensure that the inputs for time-in and time-out inputs are validated. Only allow for inputs of 00 to 23 for HH and 00 to 59 for MM. The entry for time-out should be later than time-in.
- Calculate the total number of minutes worked by each temporary staff
- Repeat this for a total of five temporary staff
- Calculate the average number of minutes worked rounded to 1 decimal place.
- Display this on the screen. Your output **must** look like this:

```
Enter name of staff: Andy
Time-in HH:MM for Andy: 08:05
Time-out HH:MM for Andy: 13:55
Enter name of staff: Ben
Time-in HH:MM for Ben: 07:03
Time-out HH:MM for Ben: 07:00
Invalid! Time-out HH:MM for Ben: eight o'clock
Invalid! Time-out HH:MM for Ben: 08:00
Enter name of staff: Charles
Time-in HH:MM for Charles: 10:03
Time-out HH:MM for Charles: 13:115
Invalid! Time-out HH:MM for Charles: 13:11
Enter name of staff: Dominic
Time-in HH:MM for Dominic: 09-04
Invalid! Time-in HH:MM for Dominic: 09:04
Time-out HH:MM for Dominic: 15:35
Enter name of staff: Ethan
Time-in HH:MM for Ethan: 08:42
Time-out HH:MM for Ethan: 16:55

Andy worked for 350 minutes
Ben worked for 57 minutes
Charles worked for 188 minutes
Dominic worked for 391 minutes
Ethan worked for 493 minutes

Average number of minutes worked: 295.8
```

11 Write your program and test that it works.

Save your program as **TPSTAFF**_<your name>_<index number>**.py**

[12]

12 When your program is complete, use the following test data to show your test result:

```
Andy
08:05
13:55
Ben
07:03
07:00
eight o'clock
08:00
Charles
10:03
13:115
13:11
Dominic
09-04
09:04
15:35
Ethan
08:42
16:55
```

Take a screen shot of your results and save it as bitmap **RESULT**_<your name>_<index number> [2]

13 Save your program as **TPSTAFF2**_<your name>_<index number>**.py**

The temporary staff are paid accordingly based on the following:

- $3 for every 15 minutes of work up to the 240th minute (4th hour)
- $4 for every subsequent 15 minutes of work after the 240th minute

Extend your program to calculate the daily wage for each staff.

Your output should look like this:

```
Andy will be paid $76
Ben will be paid $9
Charles will be paid $36
Dominic will be paid $88
Ethan will be paid $112
```

[3]

Save your program.

14 Save your program as **TPSTAFF3**_<your name>_<index number>**.py**

Extend your program to work for any number of temporary staff. The program will terminate and outputs when the user enters an empty string for the name. [3]

Save your program.

End of Paper