JUNYUAN SECONDARY SCHOOL
PRELIMINARY EXAMINATION 2020
SECONDARY FOUR EXPRESS

CANDIDATE NAME

CLASS | | | | INDEX NUMBER | |

COMPUTING

7155/02

Paper 2 (Practical)

1 Sep 2020

2 hours 30 minutes

Additional Materials: 1 x Thumb Drive
Electronic version of TASK1.xls file
Electronic version of TASK2.py file
Electronic version of TASK3.py file
Insert Quick Reference Glossary

READ THESE INSTRUCTIONS FIRST

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.

For Examiner's Use		
Task 1		10
Task 2		10
Task 3		10
Task 4		20
Total		50

Task 1

Thorntastic Durian Imports uses a spreadsheet to track its durian sales records. You are required to finish setting up the spreadsheet to record the monthly sales data for the month of March.

Open the file **TASK1.xlsx**. You will see the following data.

Some of the rows in the file have been hidden in the display below.

Thorntastic Durian Imports
11 Thorny Street, Singapore 123456

a) Durian Sales for March

March Durian Sales

Order Code	Durian Code	Quantity Ordered (KG)	Total Sales before Discount ($)	Country Code	Surcharge /Discount	Actual Sales ($)
IM-RED5529-MY		502				
IM-D241476-OT		509				
LC-D247584-SG		847				
LC-TEK5567-SG		436				
IM-MSW1863-OT		269				
IM-RED4387-OT		253				
IM-TEK1339-OT		193				
IM-MSW1357-OT		555				
IM-MSW5958-MY		122				
IM-D133956-MY		535				
LC-RED1197-SG		270				

Durian Price

Durian Code	Common Name	Cost per KG
MSW	Mao Shan Wang	$21.88
RED	Red Prawn	$15.88
D24	Sultan	$13.88
D13	Kampung	$11.88
TEK	Tekka	$17.88

Save the file as **DURIAN**_<Class>_<Index>_<Name>**.xlsx**.

1 **Figure 1** below shows an example of an **Order Code**.

The **Order Code** is made up of 13 characters and the breakdown of the code is shown below.

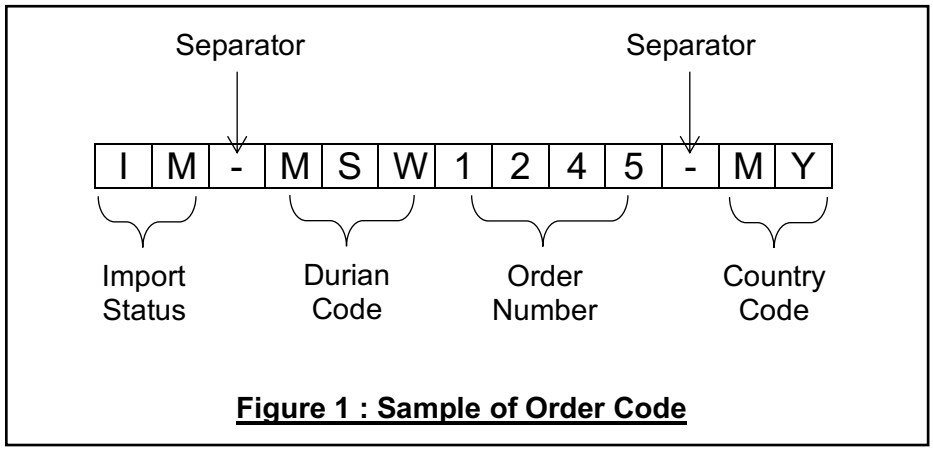

Figure 1 : Sample of Order Code

For the cell range **C7:C35**, use an appropriate function to extract the **Durian Code** from the **Order Code**. [2]

2 Use an appropriate function to search for the **Cost per KG** in the **Durian Price** table given and use it to complete the **Total Sales before Discount ($)** column. The sales must take into consideration the **Quantity Ordered (KG)**. [3]

3 For the cell range **F7:F35**, use an appropriate function to extract the **Country Code** from the **Order Code**. [1]

4 Use a conditional statement to identify orders that qualify for a discount or incur a surcharge. If the country code for the order is **SG** or **MY**, it qualifies for a **Discount**. If the country code for the order is **OT**, it incurs a **Surcharge**. [2]

5 Orders that qualify for a discount are given 12% off the total sales while orders that incur a surcharge have to pay 20% more than their total sales.
Use a conditional statement to calculate the actual sale of each order in the **Actual Sales ($)** column. [2]

Save your work.

Task 2

The following program accepts the age in years for five employees and prints out the oldest age.

```
oldest = 0
counter = 0

while counter < 5:
    age = int(input("Enter an employee's age in years: "))
    if age > oldest:
        oldest = age
    counter += 1

print("The oldest employee is ", oldest, "years old.")
```

Open the file **TASK2.py**

Save the file as **VARAGE1**_<Class>_<Index>_<Name>**.py**

6 Edit the program so that it:

(a) Accepts the age of 10 employees. [1]

(b) Prints out the oldest age as well as the average age of all employees, rounded down to a whole number. [4]

(c) Tests if the age is between 21 and 65 inclusive, and if not, asks the user for another input. [3]

7 Edit your program so that it works for any number of employees. [2]

Save your program **VARAGE2**_<Class>_<Index>_<Name>**.py**

Task 3

A number is said to be an Armstrong number when the sum of its digits raised to the power of the length of the number becomes equal to the number itself.

For example, 371 and 1634 are Armstrong numbers as follows:
$3^3 + 7^3 + 1^3 = 371$
$1^4 + 6^4 + 3^4 + 4^4 = 1634$

A computing student wrote the following program to display the list of Armstrong numbers from a user-defined range of numbers. However, there are syntax and logical errors in the program.

```
result = []

print("***********The search for ARMSTRONG numbers!***********")
limit = int(input("Range is 1 to n inclusive. \nState your n: ")
print("Checking for ARMSTRONG numbers from 1 to {}".format(limit))

for number in range(1, limit+1)
    power = len(number)
    check = number
    remainder = 0

    while check != 0:
        remainder = number%10
        sum_digits += remainder*power
        check = check%10

    if sum_digits > number:
    result.append(number)

    print("The ARMSTRONG numbers are {}:".format(result))

##List of ARMSTRONG numbers
##1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407,
##1634, 8208, 9474, 54748, 92727, 93084, 548834, ...
```

Open the file **TASK3.py**
Save the file as **ARM**_<Class>_<Index>_<Name>.

8 Identify and correct the errors in the program so that it works correctly according to the rules given. Save your program. [10]

Task 4

You have been asked to write a program to simulate the Atbash Cipher.

The Atbash Cipher is a substitution cipher with a specific key where the letters of the alphabet are reversed; i.e. all 'A's are replaced with 'Z's, all 'B's are replaced with 'Y's, and so on. It was originally used for the Hebrew alphabet, but can be used for any alphabet.

The table below shows how the cipher works for the Latin alphabet.

Plain	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Cipher	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

The plaintext of “HELLO” will be encrypted as “SVOOL” based on the cipher.
The program should allow you:

- To enter a plaintext in uppercase or lowercase.
- To encrypt the plaintext.
- Not to encrypt non-letter characters.
- To display the encrypted message on the screen. Your output **must** look like this:

```
Please enter your plaintext: Hello World!
Encrypted message: SVOOL DLIOW!
```

9 Write your program and test that it works. [12]

Save your program as **CIPHER1**_<Class>_<Index>_<Name>.

10 When your program is working, use the following plaintexts as test data to show your test results:

Test case	Expected Output
`Hello World!`	`Encrypted message: SVOOL DLIOW!`
`JUNYUAN SEC`	`Encrypted message: QFMBFZM HVX`
`ice-CREAM rules`	`Encrypted message: RXV-XIVZN IFOVH`

Take a screen shot of your results and save it as a bitmap **CIPHERRESULT**_<Class>_<Index>_<Name>. [3]

11 Save your program as **CIPHER2**_<Class>_<Index>_<Name>.

Extend your program to also allow user to choose to do a double encryption. [5]

A double encryption will flip the original encrypted message and add '%' after every two characters. A sample output could look like this:

```
Enter 1 for Single-level encryption
   or 2 for Double-level encryption: 2
Please enter your plaintext: hello world!
Encrypted message: !W%OI%LD% L%OO%VS%
```

Save your program.

End of Paper

BLANK PAGE