MINISTRY OF EDUCATION, SINGAPORE
in collaboration with
CAMBRIDGE ASSESSMENT INTERNATIONAL EDUCATION
General Certificate of Education Ordinary Level

COMPUTING

7155/02

Paper 2 Lab-based

For examination from 2025

SPECIMEN PAPER

2 hours 30 minutes

Additional Materials:	Electronic version of `CAPITALS.ipynb` data file
	Electronic version of `MATERIALS.xlsx` data file
	Electronic version of `RICEBAGS.ipynb` file
	Electronic version of `ROLLER.ipynb` file
	Insert Quick Reference Guide for Python

READ THESE INSTRUCTIONS FIRST

DO **NOT** WRITE ON ANY BARCODES.

Approved calculators are allowed.

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 70.

This document consists of **10** printed pages and **2** blank pages.

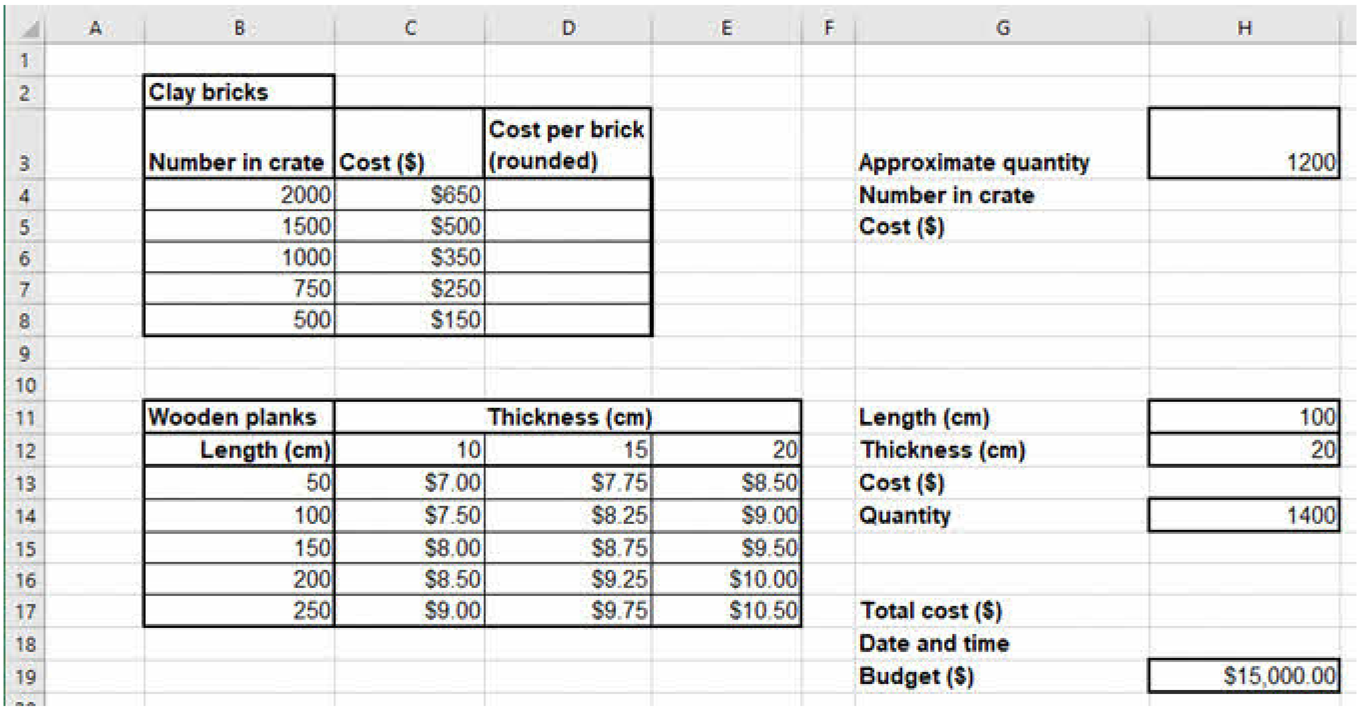
Singapore Examinations and Assessment Board

Cambridge Assessment International Education

Task 1

A company sells building materials. The company uses spreadsheet software to calculate the total cost of the building materials requested by a customer.

Open the file **`MATERIALS.xlsx`** and you will see the following data:



	A	B	C	D	E	F	G	H
1								
2		Clay bricks						
3		Number in crate	Cost ($)	Cost per brick (rounded)			Approximate quantity	1200
4		2000	$650				Number in crate	
5		1500	$500				Cost ($)	
6		1000	$350					
7		750	$250					
8		500	$150					
9								
10								
11		Wooden planks	Thickness (cm)				Length (cm)	100
12		Length (cm)	10	15	20		Thickness (cm)	20
13		50	$7.00	$7.75	$8.50		Cost ($)	
14		100	$7.50	$8.25	$9.00		Quantity	1400
15		150	$8.00	$8.75	$9.50			
16		200	$8.50	$9.25	$10.00			
17		250	$9.00	$9.75	$10.50		Total cost ($)	
18							Date and time	
19							Budget ($)	$15,000.00

The task is to finish setting up the spreadsheet to calculate the total cost of the building materials based on inputs provided by the customer in cells **H3**, **H11**, **H12**, **H14** and **H19**.

Save the file as:

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

1 In cell **D4**, enter a formula to calculate the cost per brick (rounded to the nearest $0.10) for row 4. Copy the formula into **D5:D8** to calculate the **cost per brick (rounded)** for the remaining rows. [2]

2 In cell **H18** enter a function that displays the current date and time. [1]

3 In cell **H4** enter a formula that uses a function to determine the **number in crate** from the **clay bricks** table. The **number in crate** is based on the **approximate quantity**.

The number in crate selected should be the lowest value that is above or equal to the approximate quantity required. Therefore, if an **approximate quantity** of 1200 is entered, the value selected for **number in crate** is 1500. [3]

4 In cell **H5** enter a formula that uses a function to find the **cost** of the crate of bricks from the **clay bricks** table. The **cost** is based on the **number in crate**. [2]

5 In cell **H13** enter a formula that uses functions to find the **cost** of a wooden plank from the **wooden planks** table. The **cost** is based on the **length** and **thickness** of the wooden plank. [4]

6 In cell **H17** enter a formula to calculate the **total cost** for the customer. The **total cost** is the sum of the **cost** of the clay bricks and the **cost** of the wooden planks (based on the **quantity** required). [1]

7 In cells **D4** to **D8** use conditional formatting to make the background yellow for the cell with the highest cost per brick (rounded). [2]

8 Save your work.

Instruction to candidates:

Your program code and output for each of Tasks 2 to 5 should be saved in a single `.ipynb` file using JupyterLab. For example, your program code and output for Task 2 should be saved as:

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

Make sure that each of your `.ipynb` files shows the required output in JupyterLab.

Task 2

Open the file **`RICEBAGS.ipynb`**

Save the file as:

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

The task is to edit program code that allows weights of bags of rice to be input and determines whether the bags of rice are the correct weight.

For each of the sub-tasks, add a comment using the hash symbol '#' at the beginning of your code to indicate the sub-task that the program code belongs to. For example:

```
In [1]:  # Task 2.1
         Program code
         Output:
```

Task 2.1

The following program allows the weights of 15 bags of rice to be input. The correct weight for each bag of rice must be between 4.9 kg and 5.1 kg inclusive.

```
bags_rice = 15
upper_bound = 5.1
lower_bound = 4.9
for count in range(bags_rice):
    bag_weight = float(input("Enter the weight of the bag of rice "))
    if bag weight > upper_bound:
        print("The bag of rice is overweight")
    if bag_weight < lower_bound:
        print("The bag of rice is underweight")
```

Edit the program so that it:

(a) accepts the weights for only 10 bags of rice. [1]

(b) prints out the message `"The bag of rice is the correct weight"` when a weight entered is between 4.9 kg and 5.1 kg inclusive. [2]

(c) prints out the number of bags of rice that were underweight, as well as the number that were overweight, after the weights of all the bags have been entered.

Suitable output messages must be used. [5]

Save your program.

Task 2.2

Copy and paste your program from sub-task 2.1.

Edit your program so that it works for any number of bags of rice. [2]

Save your JupyterLab notebook for Task 2.

Task 3

Open the file **`CAPITALS.ipynb`**

Save the file as:

```
TASK3_<your name>_<centre number>_<index number>.ipynb
```

The task is to edit program code so that countries and their capital cities can be added to or removed from a dictionary.

The following program has a dictionary that contains countries and their capital cities. The program allows the user to:

- input a country
- input whether they want to remove a country and its capital city from the dictionary
- input whether they want to add a country and its capital city to the dictionary.

```
capital_cities = {
    'singapore':'Singapore',
    'japan':'Tokyo',
    'australia':'Canberra',
    'england':'London',
    'france':'Paris',
    'germany':'Berlin'
}
country = input("Please enter the name of a country: ")
remove = input("Would you like to remove any of the entries? (Y or N): ")
add = input("Would you like to add a new entry? (Y or N): ")
```

For each of the sub-tasks, add a comment using the hash symbol '#' at the beginning of your code to indicate the sub-task that the program code belongs to. For example:

```
In [1]:  # Task 3.1
         Program code
         Output:
```

For all sub-tasks, you can assume that all user input is valid. All countries input to be searched or removed are found in the dictionary.

Task 3.1

Edit the program so that it:

- converts the input for `country` to lower case
- searches the dictionary for the country input and outputs the capital city of that country.

[3]

Save your program.

Task 3.2

Copy and paste your program from sub-task 3.1.

Edit the program so that if the user enters the value ‘Y’ for `remove`, the program:

- allows the user to input a country they want to remove from the dictionary
- converts the country input to lower case
- removes the country from the dictionary that is input by the user.

[3]

Save your program.

Task 3.3

Copy and paste your program from sub-task 3.2 .

Edit the program so that if the user enters the value ‘Y’ for `add`, the program:

- allows the user to input a country they want to add to the dictionary
- allows the user to input the capital city for the country they want to add
- adds the country and its capital city to the dictionary in the format `country:capital`
- outputs the dictionary at the end of the program.

[4]

Save your JupyterLab notebook for Task 3.

Task 4

Open the file **`ROLLER.ipynb`**

Save the file as:

```
TASK 4_<your name>_<centre number>_<index number>.ipynb
```

The task is to identify and correct errors in program code so that the program works according to the given rules.

The following program should check who can ride the roller coaster using the following rules:

- age over seven years
- age not more than 70 years
- height greater than 1.3 metres.

The program calculates the number of people who ride the roller coaster and the number rejected. The program finishes when an age of zero or a height of zero is input. The number of people who ride the roller coaster and the number of people rejected are then printed out.

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

```
age = 0
height = float(0)
rejected = 100
rider = 0
age = int(input("Please enter your age ))
height = float(input("Please enter your height "))
while age <> 0 and height != 0:
    if age < 7 or age > 70 or height <= 1.3:
        if age < 7:
            print("You are too young to ride")
        if age > 90:
            print("You are too old to ride")
        if height <= 1.3:
            print("You are too short to ride")
        rejected = rejected - 1
else:
        print("You can ride the Roller Coaster")
        rider = Rider + 1
    age = int(input("Please enter your age "))
    height = float(input("Please enter your height "))
print("Number of people rejected ", rider)
print("Number of riders ", rejected)
```

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

Save your JupyterLab notebook for Task 4.

BLANK PAGE

Task 5

Open a new JupyterLab notebook and save it as:

```
TASK5_<your name>_<centre number>_<index number>.ipynb
```

The task is to write a program that encrypts and decrypts messages.

A cipher is an algorithm that encodes a message so that only the intended recipient is able to read it. The Caesar Shift Cipher is a type of cipher where each letter in the original message is shifted down the alphabet sequence by a fixed number of positions.

The table below shows an example where the letters have been shifted down the alphabet sequence by 3 positions. The letters 'wrap-around' at the end. Using the table, the message 'happy' would be encoded as 'kdssb'.

Original	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
Shifted down by 3	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c

The process of encoding the message is called encryption and the encoded message is called the ciphertext. In order to read it, the recipients need to perform an opposite process called decryption to change the ciphertext back to the original message.

For each of the sub-tasks, add a comment using the hash symbol '#' at the beginning of your code to indicate the sub-task that the program code belongs to. For example:

```
In [1]:  # Task 5.1
         Program code

         Output:
```

All code should have appropriate comments and all identifiers should be appropriately named. [4]

Task 5.1

Write a `shift()` function that has the parameter `char` passed to it. The function must shift a lower-case letter down the alphabet sequence by one position (a → b ... → y → z → a) and do nothing to other characters. [4]

Task 5.2

Write a function `encrypt()` that has the parameters `message` and `positions` passed to it. The function must use the `shift()` function to encrypt the `message` argument by shifting all the lower-case letters in the message down the alphabet sequence by the number of positions given in the `positions` argument. The function should ignore all other characters. [7]

Task 5.3

Write a function `shift_up()` that has the parameter `char` passed to it. The function must shift a lower-case letter **up** the alphabet sequence by one position (a ← b ... ← y ← z ← a) and do nothing to other characters. [2]

Task 5.4

Write a function `decrypt()` that has the parameters `ciphertext` and `positions` passed to it. The function must use the `shift_up()` function to decrypt the `ciphertext` argument by shifting all the letters **up** the alphabet sequence by the number of `positions` given in the positions argument. The function should ignore all other characters. [1]

Task 5.5

Create a simple text-based user interface to:

- request the user to enter 'E' to encrypt a message or 'D' to decrypt a ciphertext (case insensitive) and to re-enter if the input is not 'E', 'D', 'e' or 'd'
- request the user to enter the message or the ciphertext
- request the user to enter the number of positions to shift the letters and the user to re-enter the number if the input is not a positive integer
- output the encrypted message and write the encrypted message to the file `encrypted.txt`, if the user requested to encrypt a message
- output the decrypted ciphertext if the user requested to decrypt a message.

Your program should use the `encrypt()` and `decrypt()` functions.

Test your program with the following input:

a, E, This is the seCret me55age!, −1, 12 [7]

Save your JupyterLab notebook for Task 5.

BLANK PAGE