# ZHONGHUA SECONDARY SCHOOL

PRELIMINARY EXAMINATION 2019
SECONDARY 4 EXPRESS

| Candidate's Name | Class | Register Number |
|---|---|---|
| | | |

# COMPUTING

# 7155/02

Paper 2 (Lab Based)

4 September 2019
2 hours 30 minutes

Additional Materials: Instruction Set

**READ THESE INSTRUCTIONS FIRST**

Write your index number and name on all the work you hand in.
Write in dark blue or black pen on both sides of the paper.
You may use a pencil for any diagrams or graphs.
Do not use staples, paper clips, glue or correction fluid.

Answer **all** the questions.
Write your answers in this question booklet.
Give non-exact numerical answers correct to 3 significant figures, or 1 decimal place in the case of angles in degrees, unless a different level of accuracy is specified in the question.
The use of a scientific calculator is expected, where appropriate.
You are reminded of the need for clear presentation in your answers.

At the end of the presentation, fasten all your work securely together.
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: |
|---|
| |

Setter: Mr. Calvin Heng
Vetter: Mr. Low Kee Ley

This question paper consists of **7** printed pages including this cover page.

## Task 1

A Server Farm Service uses spreadsheet software to calculate the cost of the service for each customer. The Server Farm has five classes of server type and offers half service and full service. You are required to finish setting up the spreadsheet to calculate the cost of each service.

Open the file **SERVERFARM.xlsx**

Save the file as **SERVERFARMCOST** _<YOUR NAME>_<INDEX NO>

**1** In cell C20 enter a formula to count the number of half service servers that have been booked. [1]

**2** In cell C21 enter a formula to count the number of full service servers that have been booked. [1]

**3** Use an appropriate function to search for the **Cost per Server class** in the **Server Class** table, and use it to complete the **Cost** column. The cost must take into consideration the **Number of Units** each customer will use. [2]

**4** Use a conditional statement to identify whether a customer will receive a **Discount** (reduction in cost). For customers that have booked more than 5 units or have bought a service that costs $500 or more, put **Yes** in the **Discount** column, otherwise put **No** in the **Discount** column. [2]

**5** Use a conditional statement to calculate the total cost of the service in the **Total Cost** column. Customers who are identified as **Yes** for the **Discount** must have a 10% reduction for the total cost. [2]

The **Total Cost** for customers without a discount is the same as **Cost**.

**6** In cell **H20** enter a formula to calculate the average total cost of all the services, rounded to the nearest dollar($). [2]

Save and close your file.

**Task 2**

The following program accepts temperature readings for 6 intervals. It also records the highest temperature reading and calculates for the average temperature reading.

Open the file **TEMPERATURE_READING.py**

Save the file as **TEMPERATURE_RECORDING_EVENT** _<YOUR NAME>_<INDEX NO>**.py**

**7** Edit the program so that it:

(a) Accepts 12 intervals, as the situation requires a more accurate picture of temperature changes. [1]

(b) Print out the highest temperature reading as well as the average temperature reading. [2]

(c) Test if the temperature reading is between -4 and 55, if not, ask the user for input again as necessary. [3]

Save your program.

**8** Save your program as **Override** _<Your _Name> _<Index_No>

**9** Edit your program so that a check is made by function emergency_override. If the function returns True, turn on emergencyFlag and break out of the for loop. If emergencyFlag is True, call function shutdown and end the program. [4]

Save your program.

**Task 3**

The following program is a Body Mass Index (BMI) Calculator that prints a medical advisory depending on the BMI value.

The program follows these rules:

- The height is entered in centimeters (whole number).
- The weight is entered in kilograms (whole number).
- The medical advisory chosen depends on these rules:

| **Range of BMI values** | **Medical Advisory** |
|---|---|
| BMI < 18.5 | Possible nutritional deficiency |
| 18.5 <= BMI < 23 | Healthy Range |
| 23 <= BMI < 27.5 | Moderate risk of diabetes and heart disease |
| BMI >= 27.5 | High risk of diabetes and heart disease |

The program prints out the BMI value and the medical advisory.

There are several syntax and logic errors in the program.

```
# BMI calculator with medical advisory

med_advisory = ['Please see a doctor for possible nutritional deficiency'
               'BMI is in healthy range! Keep up the good work with diet and exercise',
               'Please see a doctor for high risk in diabetes and heart disease',
               'Please see a doctor for moderate risk in diabetes and heart disease']

# Get User to Input height and weight

height = int(input('Please enter your height in centimetres: '))

weight = input('Please enter your weight in kilograms: ')

# Calculate BMI
# Formula to calculate BMI is:   BMI = Weight (kg) / squared(Height (m))

bmi = round((weight / (h x h)),2)

# Calculate medical advisory index based on:
# bmi < 18.5           indicates possible nutritional deficiency
# 18.5 <= bmi < 23     indicates healthy range
# 23 <= bmi < 27.5     indicates moderate risk in diabetes and heart disease
# bmi >= 27.5          indicates high risk in diabetes and heart disease

index = Null

if bmi < 18.5:

   index = 1
```

```
elif bmi < 23:

   index = 2

elif bmi < 27.5:

    index = 2

elif bmi > 27.5:

    index = 3

# Display bmi and medical advisory

print("Your bmi is " + bmi)

print("Medical Advisory " + med_advisory[idx])
```

Open the file **BMI.py**

Save the file as **BMI_AND_MEDADVISORY** _<YOUR NAME>_<INDEX NO>.**py**

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

Save your program.

**Task 4**

You have been asked to create a Multiplication Quiz Game program.

The program should:

- Test the user on the 3 (Three) times table.
- A label announcing the quiz will appear first:
  - “Welcome to the 3 Times Table Quiz”.
- Followed by the first question, which will quiz the student on any multiple from 1 to 12. For example:
  - “What is 3 x [1..12] ?”
- The next line will have a label for the student to enter his answer:
  - “Your Answer: “
- If the answer is correct, a positive label is printed, otherwise, a negative label is printed. For example:
  - “Your answer is correct! Good Job!!”
  - “Your answer is wrong! Try harder!!”
- For correct answers, 1 point is scored. Wrong answers receive zero score.
- At the end of 3 (Three) tries, the final score is printed. For example:
  - “Your score is 3 points” {For all correct answers}
  - “Your score is 2 points” {For getting 2 out of 3 correct}
  - “Your score is 1 point” {For getting 1 out of 3 correct}
  - “Your score is 0 points” {For getting none correct}

**11** Write your program and test that it works. [10]

Save your program as **MULTIPLY03_**<your name>_<index number>**.py**

**12** When your program is complete, test it for the following:

- Test 1 - Player gets all correct answers
- Test 2 - Player gets 2 out of 3 correct
- Test 3 - Player gets 1 out of 3 correct
- Test 4 - Player gets none correct [4]

Take a screenshot of Test 1, 2, 3 and 4. Save them as a **single** screenshot accordingly:
**TEST1_**<your name>_<index number>
**TEST2_**<your name>_<index number>
**TEST3_**<your name>_<index number>
**TEST4_**<your name>_<index number>

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

**13** Save your program as **MULTIPLY0212_**<your name>_<index number>**.py**

Extend your program to allow the player to choose which multiplication table to test for. Ensure that only the 2 (two) to 12 (twelve) times table can be chosen for each quiz set of 3 (three) attempts. [2]

Save your program.

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

Extend your program such that at the end of the quiz set, **after** the final score is printed, the player will receive feedback lines for each of the attempts. For example (A quiz set on the 4 (four) times table): [4]

- “What is 4 x 3?  Your answer was 12 [CORRECT]
- “What is 4 x 10? Your answer was 40 [CORRECT]
- “What is 4 x 4? Your answer was 15 [WRONG]

Save your program.

***End of Paper***