Bukit View Secondary School

Secondary Four Express
Preliminary Examination 2020

CANDIDATE NAME

CLASS

INDEX NUMBER

COMPUTING **7155/02**

Paper 2 (Lab-based) **16 Sep 2020**

2 hours 30 minutes

Additional Materials : Electronic version of `SALES.xlsx` data file
Electronic version of `NUMCAT.py` file
Electronic version of `DATES.py` file
Insert Quick Reference Glossary

READ THESE INSTRUCTIONS FIRST

Do not open this booklet until you are told to do so.

Write your candidate name, class and number in the spaces at the top of this page.
Write in dark blue or black pen.

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
Marks
50

Setter: Mrs Leong Wee Choo **Parent's Signature:__________________**

This question paper consists of **9** printed pages.

Task 1

Jane uses an online selling platform to sell women's bags and uses a spreadsheet to keep track of her monthly sales.

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

	A	B	C	D	E	F	G
1	**Sales**						
2	**Period**	**Category**	**Quantity Sold**	**Sales**	**Hit target?**		
3	January 2019	ToteBag	27				
4		Backpack	26				
5		Wallet	47				
6		SlingBag	36				
7		CoinPouch	11				
8		Clutch	5				
9							
10		Total monthly sales					
11		Number of category exceeding target					
12							
13	**Selling Price Per Cateogry**						
14	**Category**	**Backpack**	**Clutch**	**CoinPouch**	**SlingBag**	**ToteBag**	**Wallet**
15	**Selling price per unit**	$ 25.90	$ 15.00	$ 12.90	$ 16.60	$ 35.90	$ 55.00
16							
17	Projected yearly profit						
18							
19	Amount to repay bank						
20							
21	She is		to repay the bank after 3 years.				

Save the file as **SALES**_<your name>_<index number>.xlsx

1 The **Sales** need to be calculated for the **Quantity Sold**.

In cells D3 to D8 enter a formula that uses an appropriate function to search for the **Selling price per unit** in the **Selling Price Per Category** table. Use it to calculate the **Sales** of each category. [2]

2 In cell D10 enter a formula to calculate the total sales. [1]

3 In cells E3 to E8 enter a conditional statement to identify whether the target sales has been hit. The target sales for each category has been set to $500. For category that hit the target sales put **Yes** in the **Hit target?** column, otherwise put **No** in the **Hit target?** column. [1]

4 In cell D11 enter a formula to count the number of categories that exceed the sales target. [1]

5 In cell B17 enter a formula to calculate her **Projected yearly profit** if her monthly profit is 20% of her total monthly sales. You can assume that her sales per month remains fairly constant. [2]

6 She took up a loan of $30000 to start her business. The bank charges her 3.88% per annum compounded monthly for a period of 3 years.

In cell B19 enter an appropriate function to calculate the amount she needs to pay after 3 years. [2]

7 In cell B21 enter a conditional statement to determine if she is able to pay the bank based on her projected yearly profit after 3 years by putting **able** or **unable**. [1]

Save and close your file.

Task 2 begins on the next page.

Task 2

The following program allows 10 integers to be input. The numbers which must be within 1 to 100 inclusive are then printed as Low, Medium or High.

```
for count in range(10):
    number = int(input('Enter a positive integer: '))

    if number > 0 and number <= 30:
        print(number, '- Low')
    elif number > 30 and number <= 70:
        print(number, '- Medium')
    elif number > 70 and number <= 100:
        print(number, '- High')
```

Open the file **NUMCAT.py**

Save the file as **NUMCATV1**_<your name>_<index number>**.py**

8 **(a)** Edit the program so that it will accept any number of input. The program will end only when the user enters 0. You can assume that only digits will be entered. [2]

(b) Edit the program to:

- test whether the user has entered a number in the correct range
- output a suitable error message that asks the user to enter the number again, and repeat this until the user enters a valid number. [2]

(c) Edit the program to print how many numbers were classified as “Low”, “Medium” or “High”, after the user entered 0.

A sample output is shown below. [3]

```
There are 1 number(s) in Low category
There are 2 number(s) in Medium category
There are 2 number(s) in High category
```

Save your program.

9 Save your program as **NUMCATV2_**<your name>_<index number>**.py**

Valid numbers are now defined as numbers within 1 to `upper_limit` inclusive, where `upper_limit` is a user input. You need not validate `upper_limit` and can assume it will be a number more than 71.

As a result of this change, numbers that are greater than 70 but smaller than or equal to `upper_limit` will be categorized as High.

Edit your program so that it works correctly with this new boundary. [3]

Save your program.

Task 3 begins on the next page.

Task 3

The following program reads in 3 numbers, each a part of a date and the program will print the date in the format dd Mmm yyyy.

For example, user enters 9, 4, 1991. The program will print 09 Apr 1991.

The program uses the rules:

- there are 30 days in the months of April, June, September and November
- if day entered is less than 10, the output will have a '0' prepended
- there is a space between the day, month and year
- year should be between 1900 and 2200 inclusive

This program assumes non-leap year and all inputs are digits.

There are several syntax and logic errors in the program.

```
month_abbr = ['Jan','Feb','Mar','Apr','May','Jun',
              'Jul','Aug','Sep','Oct','Nov','Dec']

day = int(input("Enter the day: "))

# validate day #
while day < 1 or day > 30:
    day = int(input("Invalid day. Re-enter the day: "))

month = int(input("Enter the month: "))

#validate month
while month < 1 and month > 12:
    month = int(input("Invalid month. Re-enter the month: "))

year = int(input("Enter the year: "))

# validate year
while year > 1900 or year > 2200:
    year = int(input("Invalid year. Re-enter the year: "))

if day > 28 and month = 2:
    print('Invalid date!')
elif month in [4, 7, 9, 11] and day > 30:
    print('Invalid date!')
else:
    if day < 10:
        day = 0 + day
    answer = str(day)+' '+month_abbr[month]+' '+str(year)
print(answer)
```

Open the file **DATES.py**

Save the file as **MYDATES**_<your name>_<index number>**.py**

10 Identify and correct the errors in the program. [10]

Save your program.

Task 4 begins on the next page.

Task 4

You have been asked to create a code-breaking game for two players.

The program must:

- allow player 1 to input a 4-character code denoted by the first letter of these colours : **R**ed, **B**lue, **G**reen, **Y**ellow, **P**urple, **O**range. The code will be converted to upper case by the program after entry. The program must ask for another code each time the player enters a code that is not 4-character long or comprises of letters not from the first letter of the above 6 colours.
- allow player 2 to enter a 4-character capitalized code. Player 2 will have 4 guesses to correctly guess the code input by player 1. You do not need to validate the input for player 2.
- Output "Code broken!" when player 2 inputs the same code as player 1 and the game must end.
- Output "Incorrect" when player 2 does not input the same code as player 1.
- Output "Game over!" when player 2 has entered four incorrect codes.

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

Save your program as **CODEBREAKER1**_<your name>_<index number>.**py**

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

- Test 1 – Player 1 inputs `RBP`
- Test 2 – Player 1 inputs `OIBR`
- Test 3 – Player 1 inputs `GYBG` and
  Player 2 inputs `OBGY, RBGY, BBGY, RBBY`
- Test 4 – Player 1 inputs `GYBY` and
  Player 2 inputs `GYYY, GYBY`

Take a screenshot of: [4]

- Test 1, 2 and 3. Save this **single** screenshot as:
  TEST123_<your name>_<index number>

- Test 4. Save this screenshot as:
  TEST4_<your name>_<index number>

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

13 Save your program as **CODEBREAKER2**_<your name>_<index number>.**py**

Extend your program to:

- If the guess is wrong, output a 4-character string indicating “Y” if the position and colour are correct or “N” otherwise. For instance, if player 1 inputs “ROBY” and player 2 inputs “RPYB”, “YNNN” should be output.
- If player 2 inputs a correct code, the program **must** output the number of guesses made. [5]

Save your program.