Bukit View Secondary School

Secondary Four Express
Preliminary Examination 2021

CANDIDATE NAME

CLASS

INDEX NUMBER

COMPUTING

Paper 2 (Lab-based)

7155/02
14 September 2021
2 hours 30 minutes

Additional Materials: Electronic version of `STUDYLOAN.xlsx` data file
Electronic version of `SCORES.py` file
Electronic version of `CHECKDIGIT.py` file
Insert Quick Reference Glossary

READ THESE INSTRUCTIONS FIRST

Write your name, class and index 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: Ms Patricia Lye

Parent's Signature:__________________

Task 1

Luke wants to take up a $50,000 study loan for his undergraduate studies. The loan is to be repaid over 5 years. He uses spreadsheet software to compare the repayments among 5 different banks. Interest rates stated are per annum but compounded monthly.

You are required to finish setting up the spreadsheet.
You are required to use the following format for all payments.

	A	B
1	**Monthly Payment**	($1,234.56)
2	**Interest Payment (First month)**	($1,234.56)
3	**Total Payment**	($1,234.56)

Open the file **STUDYLOAN.xlsx** and you will see the following data.
Cells **B7** to **F7** and **B20** to **D20** are formatted as percentage.

	A	B	C	D	E	F
1	**STUDY LOAN REPAYMENT COMPARISON**					
2						
3	**Loan amount**	$50,000.00				
4	**Loan Length (months)**	60				
5						
6	**Banks**	**AMAX**	**CIMD**	**DBX**	**OCBD**	**MAIBANK**
7	**Interest Rate per annum (in percentage)**	4.85%	4.47%	4.38%	4.45%	4.50%
8	**Monthly Payment**					
9	**Interest Payment (First month)**					
10	**Processing Fee**	Mid	Mid	High	High	Low
11	**Total Payment**					
12						
13	**Range**					
14	**Mean**					
15						
16						
17						
18	**Processing Fee Rate**					
19	**Processing Fee**	**Low**	**Mid**	**High**		
20	**Rates (in percentage)**	1.50%	2.00%	2.50%		

Save the file as
MYSTUDYLOAN_2021_<your name>_<index number>_<class>.**xlsx**

1 In cells **B8** to **F8**, enter a formula that uses an appropriate function to calculate the **Monthly Payment** for each bank. [2]

2 In cells **B9** to **F9**, enter a formula that uses an appropriate function to calculate the **Interest Payment (First month)** for the first month of the loan for each bank. [2]

3 The **Total Payment** needs to be calculated. On top of paying interest, banks also charge a one-time processing fee which is charged based on the loan amount.

In cells **B11** to **F11**, enter a formula that uses an appropriate function to search for **Rates (in percentage)** in the **Processing Fee Rate** table. Use it to calculate the **Total Payment** at the end of the loan. [2]

4 In cell **B13**, enter a formula that uses an appropriate function to calculate the range of the 5 **Total Payment**. [1]

5 In cell **B14**, enter a formula that uses an appropriate function to calculate the mean of the 5 **Total Payment**. [1]

6 In cells **B6** to **F6**, use a formatting tool to change the colour of the cell to green if the bank offers the lowest total amount Luke would have to pay. [2]

Save and close your file.

Task 2 begins on the next page.

Task 2

The following program requests user to enter a score and then save the score into a list. It will keep looping until the user has no more score to enter.

```
scores = []
more_input = "Y"
while more_input == "Y":
    current_score = int(input("Please input score: "))
    scores = scores + [current_score]
    more_input = input("Do you want to enter another score? 'Y' or 'N': ")
```

Open the file **SCORES.py**.

Save the file as **MYSCORES_2021_**<your name>_<index number>_<class>**.py**

7 Edit the program so that it converts the score input by the user to a float. [1]

Save your program.

8 Edit the program to only accept a score between 0 and 100 (inclusive) to be input. A suitable error message must be displayed if the score input is not in this range. The program must loop until a valid score is input. [3]

Save your program.

9 Edit the program to only accept "Y" or "N" to be entered when asked if there is another score to be input. A suitable error message must be displayed if an incorrect input is given. The program must loop until the user inputs either "Y" or "N". [3]

Save your program.

10 Edit the program to calculate and output the mean score after all the scores have been input. The mean score, formatted to nearest whole number, should be printed together with an appropriate message. [3]

Save your program.

Task 3

A check digit is a digit or letter added to a sequence of digits to detect errors in manual input. One popular standard for check digits is the 12-digit Universal Product Code (UPC) standard. To calculate the check-digit for UPC standard, we process the first 11 digits by:

- adding the digits in the odd-numbered positions (i.e., 1st, 3rd … 11th)
- multiplying the result by three
- adding together the digits in the even-numbered positions (i.e., 2nd, 4th … 10th) to the result
- dividing the result by 10 to find the remainder
- subtracting the remainder from 10 to find check digit

If the remainder is zero (0), the check digit is 0.

The following program uses the UPC standard to calculate the check digit based on the first 11 digits entered by user. It then outputs the check digit, the number of check digits found and repeats the process until the user indicates 'N' when prompted if another check digit needs to be found.

There are several logic and syntax errors in the program.

```
repeat = "N"
count = 1
while repeat == "Y":
    upc = input("Enter the first 11 digits of UPC: )
    even_sum = 0
    odd_sum = 0

    while len(upc) != 11 and upc.isdigit() == False:
        upc = input("Your input is not 11 digits. Enter the first 11 digits of UPC: " )

    for index in range(10):
        digit = upc[index]
        if (index - 1) % 2 == 0:
            even_sum = even_sum + digit
        else:
            odd_sum = odd_sum + digit

    check_digit = (odd_sum * 3 + even_sum) % 10
    if check_digit == 0:
        check_digit = 10 - check_digit

    count = count + 1
    print("The check digit is " + check_digit)
    print("You have found " + str(count)  " check digit(s).")

    repeat = input("Do you want to find the check digit of another UPC? 'Y' or 'N'? ")
```

Open the file **CHECKDIGIT.py**.

Save the file as **MYCHECKDIGIT_2021_<**your name>_<index number>_<class>.**py**

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

Save your work.

Task 4

An anagram is a word formed by rearranging the letters of a different word, using all the original letters exactly once.

You have been asked to create an anagram checker.
The program must fulfil the following criteria:

- Allow player 1 to input a word that will be converted to lower case by the program after entry. The program must ask for another word each time the player's input is not made up of alphabets.
- Allow player 2 to enter an anagram based on player 1's word. It will be converted to lower case by the program after entry. The program must ask for another word each time the player's input is not made up of alphabets and if the length of player's 2 word differs from player's 1.
- Output "XXXXX…X" based on player 2's word. Each "X" should be replaced by the index position of player 1's word. If a letter is not found in player 1's word, then it should remain as "X".
  If a letter in player 1's word has already been used, the same letter's index cannot be used again.
  If a letter in player 1's word appears fewer times than the same letter in player 2's word, the letter should remain as "X".
  Examples:
  - Player 1 input `"State"` and player 2 input `"Taste"`.
    The output should be `"12034"`.
  - Player 1 input `"Pastel"` and player 2 input `"Spaces"`.
    The output should be `"201X4X"`.

12 Write your program and test that it works. [13]

Save your program as
ANAGRAM1_2021_<your name>_<index number>_<class> .**py**

13 When your program is complete, test it using the following inputs:

- Test 1 – Player 1 inputs word `happy123`
- Test 2 – Player 1 inputs word `State`
  Player 2 input word `Eats`
  Player 2 input word `Treat`
- Test 3 – Player 1 inputs word `Stressed`
  Player 2 input word `Desserts`

Take a screenshot of:

- Test 1 and Test 2. Save this single screenshot as **TEST12_2021_**<your name>_<index number>_<class>

- Take a screenshot of Test 3. Save this single screenshot as **TEST3**_<your name>_<index number>_<class> [3]

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

14 Save your program as **ANAGRAM2**_<your name>_<index number>_<class> .**py**

Player's 2 word is a possible valid anagram if all the letters in Player's 1 word is used exactly once.

Extend your program to:

- output a message whether player 2's word is a possible valid anagram
- prompt user to enter “Y” to restart the game or “N” to end the game after every round
- if “Y” is entered, a new game is started. [4]

Save your program.

- End of Paper -

BLANK PAGE