Task 1

Anna is considering opening a cafe. To help in her decision-making process, she uses a spreadsheet to plan out the expenditure and loan components. Complete the spreadsheet according to the following instructions.

	A	B	C	D	E
1	**Financial Plan for New Cafe Start-up**				
2					
3	**Expenditure**				
4					
5	**Item**	**Monthly Expenditure**	**Initial Expenditure**		
6	Furniture / Interior Design		20000		
7	Kitchen Equipment / Machines		50000		
8	Rental	5000	30000		
9	Staff Salary	20000			
10	Staff Training		1000		
11	Utility	800			
12		**Item Count**			
13					
14	**Loan**				
15					
16	Loan Amount:	$20,000.00			
17	Interest Rate (per annum):				
18	Instalment (per month):				
19	Total Repayment Amount:				
20					
21	**Month**	**Principal Payment**	**Interest Payment**	**Remaining Principal**	**Is Interest > 2%?**
22	1				
23	2				
24	3				
25	4				
26	5				
27	6				
28	7				
29	8				
30	9				
31	10				
32	11				
33	12				
34					
35					
36	**Interest Rates Table**				
37	**Loan Amount**	**Interest Rate (per annum)**			
38	$10,000.00	3%			
39	$50,000.00	4%			
40	$100,000.00	5%			
41					

Open the file **CAFE_STARTUP.xlsx**.

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

1 Change the format of cells **B6:C11** to currency with two decimal places. [1]

2 In cell **C12**, enter a formula to count the number of items which contribute towards the initial expenditure. [1]

3 In cell **B17**, enter a formula to search for the **Interest Rate (per annum)** in the **Interest Rates Table**. [1]

4 Let us assume that Anna would like to loan $20000 to be repaid over 12 months. In cell **B18**, enter a formula to calculate the monthly instalment. Give your answer as a positive value. [1]

5 In cell **B19**, enter a formula to calculate the total repayment amount at the end of 12 months. [1]

6 In cell **B22**, enter a formula to calculate the principal payment for the first month and use it to complete the **Principal Payment** column. Give your answer as a positive value. [2]

7 In cell **C22**, enter a formula to calculate the interest payment for the first month and use it to complete the **Interest Payment** column. Give your answer as a positive value. [1]

8 In cell **D22**, enter a formula to calculate the remaining principal after payment is made for the first month and use it to complete the **Remaining Principal** column. [1]

9 In cell **E22**, enter a conditional statement to determine if the interest payment exceeds 2% of the monthly instalment in the first month. Use **Y** for Yes and **N** for No. Complete the **Is Interest > 2%?** column. [1]

Task 2

For over 90 years, The Walt Disney Studios has brought quality movies, music and stage plays to consumers throughout the world. Three well-known Disney movies are Aladdin, Bambi and Cinderella. Five students were tasked to vote for their favourite movie among the three. The following program prompts each student to enter his/her choice and stores the vote counts in an array. After all the students have cast their votes, the program returns the title of the most popular movie and number of votes it received. Should there be a tie in the number of votes for the most popular movie, the title with the lowest array index is returned.

```
noOfStudents = 5
highestVoteCount = 0
highestVoteCountIndex = 0
movies = ["Aladdin", "Bambi", "Cinderella"]
voteCounts = [0] * len(movies)

for i in range(noOfStudents):
    choice = int(input("Cast your vote (enter 1 for Aladdin, 2 for Bambi, 3 for Cinderella): "))
    voteCounts[choice - 1] = voteCounts[choice - 1] + 1

for j in range(len(voteCounts)):
    if voteCounts[j] > highestVoteCount:
        highestVoteCount = voteCounts[j]
        highestVoteCountIndex = j

print("The most popular movie is", movies[highestVoteCountIndex], "with", highestVoteCount, "votes.")
```

Open the file **MOVIE_VOTING.py**.

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

10 Edit the program such that it:

(a) Accepts entries from ten students. You may assume that only positive whole numbers are entered. [1]

(b) Validates entries. When an invalid entry is entered, the program should continuously prompt the student to enter a valid entry with an error message. The program proceeds only after a valid entry is entered. [2]

(c) Prints the message "`Vote accepted`" when a valid entry is entered. [1]

(d) Additionally, returns the title of the least popular movie and number of votes it received. Should there be a tie in the number of votes for the least popular movie, the title with the lowest array index is returned. [2]

(e) Appends the movie “Enchanted” to the array of movies. The message prompting for entry and validation should be amended accordingly. [2]

Save the program.

11 Save the program as
VARMOVIE_<your name>_<centre number>_<index number>.py.

Edit the program such that it works for any number of students. Ensure that the number of students entered is an integer. [2]

Save the program.

Task 3

Universal Studios Singapore (USS) has a high speed turbulent thrill ride called Battlestar Galactica where riders can choose between the Cylon coaster or Human coaster each time. Let us assume that USS would like to determine the more popular coaster on a daily basis. To do so, a system was implemented to generate a unique 13-digit code for every rider at the point of entry. The codes are stored in a database and have the following structure.

X	D	D	M	M	Y	Y	H	H	M	M	S	S
Coaster indicator [1 for Cylon, 2 for Human]	Date of entry in DDMMYY format						Time of entry in HHMMSS format					

For simplicity, the following program has been preloaded with ten unique codes. The program prompts the user to enter a date of interest. It then extracts the necessary information from each code and stores the count for each coaster in an array accordingly. After all the codes have been processed, the program returns the name of the more popular coaster and number of riders for the day.

There are several logic and syntax errors in the program.

```
codes = ["1010119100123",
         "1010119151313",
         "1010119155253",
         "1010119174012",
         "2010119101546",
         "2010119113453",
         "2010119114022",
         "1020119150414",
         "2020119193511",
         "2020119194025"]
coasterCounts = [0] * 2 #Use coasterCounts[0] for Cylon coaster and coasterCounts[1] for Human coaster

dateOfInterest = int(input("Enter date of interest (in DD format): "))

for i in range(len(coasterCounts)):
    currentCode = codes[i]
    coaster = currentCode[0]
    date = int(currentCode[1:2])
    if date = dateOfInterest:
    coasterCounts[coaster + 1] = coasterCounts[coaster + 1] - 1

if coasterCounts[0] < coasterCounts[1]:
    print("The more popular coaster is Cylon with", coasterCounts[0], "riders.")
else:
    print("The more popular coaster is Human with", coasterCounts[2], "riders.")
```

Open the file **BATTLESTAR_GALACTICA.py**.

Save the file as
MYBATTLESTAR _<your name>_<centre number>_<index number>.py.

12 Correct the errors such that the program works as expected. [10]

Save the program.

Task 4

Develop a program for a two-player guessing game.

Your program should:

- Prompt Player 1 to enter three different numbers between 1 and 20 inclusive in the format X-Y-Z where X, Y and Z are three numbers.
- Validate the input by Player 1 to ensure that the three entries are numbers within the range. When an invalid entry is entered, the program continuously prompts Player 1 to re-enter the three numbers with an error message. The program proceeds only after a valid input is received.
- Prompt Player 2 to guess the three numbers in the format X-Y-Z where X, Y and Z are three numbers. For Player 2 to win, the three numbers must match those of Player 1 in the same order. You do not need to validate the input by Player 2.
- Allow Player 2 three attempts.
- Print the message "`You win`" when Player 2 wins. The game ends.
- Print the message "`Game over`" when Player 2 is unable to win after three attempts. The game ends.

13 Write your program. [10]

Save your program as
MYGAME1_<your name>_<centre number>_<index number>.py.

14 Test your completed program using the following test cases.

Case 1
Player 1 enters 5-10-15
Player 2 enters 5-10-15

Case 2
Player 1 enters 5-10-15
Player 2 enters 1-2-3
Player 2 enters 4-5-6
Player 2 enters 7-8-9

Case 3
Player 1 enters 5-10-25
Player 1 enters 5-10-15
Player 2 enters 5-10-15

Case 4
Player 1 enters A-10-15
Player 1 enters 5-10-15
Player 2 enters 5-10-15

Take an individual screenshot for each test case and save each screenshot as **TEST<case number>_<your name>_<centre number>_<index number>**.
There should be four screenshots in total. [4]

15 Save your program as
MYGAME2_<your name>_<centre number>_<index number>.py.

Extend your program as follows:

- For attempts with one or two correct entry/entries, inform Player 2 which is/are the correct entry/entries.
- For attempts with no correct entries, inform Player 2 accordingly. [2]

Save your program.

16 Save your program as
MYGAME3_<your name>_<centre number>_<index number>.py.

Extend your program as follows:

- Prompt Player 2 to enter the level of difficulty. Use 1 is for beginner, 2 for intermediate and 3 for advanced. You do not need to validate the input by Player 2.
- For beginner level, allow Player 2 five attempts.
- For intermediate level, allow Player 2 four attempts.
- For advanced level, allow Player 2 three attempts.
- If Player 2 wins, print an additional message "`X attempts used`" where X is the number of attempts utilised. [4]

Save your program.