Question 1 (15 marks)

Effective 15 December 2021, the government announced additional cooling measures for property buyers. Stamp duty is imposed on the price of the property purchased. Buyers of properties have to pay Buyer's Stamp Duty (BSD) and Additional Buyer's Stamp Duty (ABSD). BSD is imposed on any property purchased and is calculated based on the following table.

Purchase price	BSD rate
First $180,000	1%
Next $180,000	2%
Next $640,000	3%
Remaining amount	4%

Table 1-1

For example, a property bought for $1,200,000 will incur BSD of
(1% x 180,000) + (2% x 180,000) + (3% x 640,000) + (4% x 200,000) = $32,600

In addition to BSD, ABSD is imposed based on the following table.

Buyer's residency status	ABSD rate on purchase of first property	ABSD rate on purchase of second property	ABSD rate on purchase of third and subsequent property
Singapore citizens	Not applicable	17%	25%
Singapore Permanent Residents	5%	25%	30%
Foreigners	30%	30%	30%

Table 1-2

Express a sequence of statements based on computational logic and write a program that helps a buyer calculate the BSD and ABSD. The program reads in the citizenship status, current number of properties owned, price of property and displays the BSD and ABSD. Sample program executions are as follows:

Sample 1 Enter citizenship (SG, PR, FR): **SI** Invalid citizenship! **Sample 2** Enter citizenship (SG, PR, FR): **sg** Enter number of properties currently owned: **0** Enter price of property to purchase ($): **768800** Property price: $768,800.00 Citizenship: Singaporean Number of properties currently owned: 0 BSD is $17,664.00 **Sample 3** Enter citizenship (SG, PR, FR): **PR** Enter number of properties currently owned: **1** Enter price of property to purchase ($): **1,230,000** Property price: $1,230,000.00	**Comments** Citizenship must be SG, PR or FR. Accept lowercase. For price, assume input format will be integers and can include commas to separate the thousands. Therefore 1760000 or 1,760,000 are both acceptable. Assume that the commas will be put at the correct position. No need to validate that. Assume input for number of properties will be integer. Check for negative <0 and display "Number of property cannot be negative" if invalid.

Citizenship: Permanent Resident Number of properties currently owned: 1 BSD is $33,800.00 ABSD is $307,500.00	Output for citizenship should be Singaporean, Permanent Resident or Foreigner.

This question covers materials in Seminar 1 and 2. Use **selection structure** for this question. Functions, repetition or collections are not required for this question.

Paste screenshots of 2 program executions, with different input.

(15 marks)

Question 2 (25 marks)

Write a program to solve computational problems using structured programming. The program allows a user to practice addition, subtraction and multiplication of numbers.

The scope of the program is as follows:

- Allow user to input the highest level to achieve. Highest level is 5.
- Each level consists of 3 questions to add, subtract or multiply 2 numbers. The operators for the 3 questions appear in random sequence.
- Addition. For level 1, both operands consists of single digits 1 to 9. For level 2, both operands are double digits and for level 3, both operands are 3 digits and so on. All the numbers should be randomly generated.
- Subtraction. The number of digits for the operands for each level are the same as addition, except that the second operand must be less than the first, so that the result of subtraction will always be greater than 0.
- Multiplication is the same as addition in terms of the number of digits for the first operand at each level. For level 1, the first operand should not start with 1. The second operand will always be a single digit from 2 to 9 for all levels.
- The exercise starts at level 1, with questions from each of the 3 operators, but in random order.
- At the end of a level, if there are errors in the answers, the user has to repeat that level. If all the answers are correct, program proceeds to the next level.
- When the highest level is completed, the program ends.

A sample program execution is as follows:

Enter highest level to achieve (1-5): **0** Highest level must be between 1 and 5. Enter highest level to achieve (1-5): **6** Highest level must be between 1 and 5. Enter highest level to achieve (1-5): **3** Level 1 Q1. 6 - 4 = **2** Correct! Q2. 6 * 8 = **48** Correct! Q3. 5 + 4 = **9** Correct! Well done! Press <Enter> to proceed to level 2. **<Enter>** Level 2 Q1. 65 - 20 = **45** Correct! Q2. 81 + 41 = **121**	Comments Validate input. Prompt again if level is not between 1-5. Assume input will be integers. No validation required. If the answer is correct, display Correct!. If the answer is wrong, provide the answer. All 3 operators appear in a random order. Pause to allow user to press Enter key to continue next level.

Wrong! Answer is 122 Q3. 48 * 6 = **108** Wrong! Answer is 288 You have 2 errors. Press <Enter> to repeat this level. **<Enter>** Level 2 Q1. 64 - 63 = **1** Correct! Q2. 98 * 6 = **588** Correct! Q3. 41 + 73 = **114** Correct! Well done! Press <Enter> to proceed to level 3. **<Enter>** Level 3 Q1. 583 * 9 = **5247** Correct! Q2. 608 - 333 = **275** Correct! Q3. 307 + 871 = **1179** Wrong! Answer is 1178 You have 1 error. Press <Enter> to repeat this level. **<Enter>** Level 3 Q1. 583 - 296 = **287** Correct! Q2. 961 * 6 = **5766** Correct! Q3. 205 + 148 = **353** Correct! Well done! You have completed the highest level 3!	Note the display is errors, plural. Since not all the answers are correct, stay at same level. Note the display is error, singular. Quit when highest level is reached.

(a) Implement a function `getExpression(level, operator)` that has one integer parameter representing a level and another string parameter representing the arithmetic operator. The function generates a string expression, and the answer based on the requirements for the level. For example, `getExpression(2, '-')` may return '`23 - 17 = `', 6. Both the string expression and the result value are returned.

(10 marks)

(b) Apply structured programming and implement the program. Include a screenshot of a program execution.

(15 marks)

This question covers materials up to Seminar 3. Make use of functions, selection and repetition structures. **NO** data structures like sets, lists or dictionary should be used for this question. The `eval()` function should not be used. Keep the program modular by defining other functions if necessary.

Question 3 (25 marks)

This question is similar to question 2, but with multiple players competing against each other.

You are to develop an application to address practical requirements and rules as follows:

- There are multiple players. Players shall be named A, B, C and so on.
- The expressions generated are the same as Question 2, except that for a level, the same operator is used for all players.
- The operator is randomly selected for a level.
- Game starts at level 1. A player gets eliminated if the wrong answer is given. However, if all the players answer incorrectly for a level, they repeat the same level again.
- Game progresses to next level for the remaining players who answer correctly.
- The lone player who gets the correct answer is the winner.

A sample program executions is as follows:

Enter number of players: **1** Please enter at least 2 players. Enter number of players: **4** 4 players ['A', 'B', 'C', 'D'] -- Level 1 -- Player A, 1 + 6 = : **7** Correct! Player B, 2 + 1 = : **3** Correct! Player C, 3 + 3 = : **6** Correct! Player D, 1 + 2 = : **3** Correct! Players ['A', 'B', 'C', 'D'] progress to the level 2. -- Level 2 -- Player A, 25 - 19 = : **5** Incorrect! Player A eliminated. Player B, 88 - 78 = : **10** Correct! Player C, 35 - 28 = : **6** Incorrect! Player C eliminated. Player D, 22 - 17 = : **5** Correct! Players ['B', 'D'] progress to the level 3. -- Level 3 -- Player B, 632 - 398 = : **233** Incorrect! Player B eliminated. Player D, 241 - 174 = : **68** Incorrect! Player D eliminated. Players ['B', 'D'] will challenge same level 3 again. -- Level 3 -- Player B, 153 - 140 = : **13** Correct! Player D, 104 - 103 = : **1** Correct! Players ['B', 'D'] progress to the level 4. -- Level 4 -- Player B, 4781 + 2117 = : **6897** Incorrect! Player B eliminated. Player D, 2220 + 6229 = : **8449** Correct! Player D is the winner!	Comments Display the players. All operators are the same for a level. Level 1 all players correct answer, proceed to level 2. Level 2, players A, C eliminated. Players B, D proceed to next level. Since, there are no winners for the level, the players play the same level again. Both progress to next level. The lone player with the correct answer wins.

(a) Implement a function `getPlayers()` that prompts for the number of players. The number must be at least 2. The function returns the players names as a list using letters A, B, C, D etc. depending on the number of players input. A sample run of this function is as follows:

```
Enter number of players: 1
Please enter at least 2 players.
Enter number of players: 4
4 players ['A', 'B', 'C', 'D']
```

(5 marks)

(b) Employ structured programming principles to develop the program. Make use of the `getExpression()` function in question 2(a).
Paste a screenshot of a program execution that covers all scenarios.

(20 marks)

This question covers materials up to seminar 4. The data structure to use is List. You can use more than 1 list. No nested list or dictionary collection is required for this question.

Question 4 (35 marks)

Customers of a popular Nasi Briyani stall experience long queues when placing their orders. This has frustrated customers and so the owner wants a program that allows customers to place orders, issues an order number, and when the order number is called, customers can collect their orders.

Apply data structures to store and process information. The scope and assumptions for this question are as follow:

- The food stall has a menu, with item code, description and price.
- After an order is placed, the customer will be issued an order number.
- When the order is ready, the order number is called, and customer is served the order.
- Customers can update or cancel order.

The food menu is stored in a `menu.txt` file as follows:

```
B1,Chicken briyani,5.00
B2,Mutton briyani,5.50
B3,Fish briyani,6.00
```

Refer to Appendix A for the contents of this file. The program reads the file, and creates a dictionary with the following structure:

```
menu = {'B1':['Chicken briyani',5], 'B2':['Mutton briyani',5.5], 'B3':['Fish briyani', 6.0]}
```

The data structure to store orders is a dictionary. This is a sample of the dictionary after several orders:

```
orders = {1:[ ['B1', 1] ], 2:[ ['B1', 2], ['B2', 1] ] }
```

The key is the order number, and the value is a list consisting of the details of the order. The order may comprise several items, therefore, a list is used to store these items. For example, the order number 2 is a list, consisting of 2 lists of items: B1 x 2 and B2 x 1.

The program is menu driven as follows:

```
ABC Briyani
1. Place order
2. Update order
3. Serve order
0. Exit.
```

(a) Place order. This option allows a customer to place an order. The order menu is displayed and the user keys in an item code, followed by a space, followed by quantity. Several menu items may be entered. To end the order, the user simply press the `Enter` key. A sample session is as follows:

```
ABC Briyani
1. Place order
2. Update order
3. Serve order
0. Exit
Enter option: 1
Menu
B1 - Chicken briyani $5.00
B2 - Mutton briyani $5.50
B3 - Fish briyani $6.00
Enter item code and quantity (Press <Enter> key to end): B1 1
Enter item code and quantity (Press <Enter> key to end): B4 2
Invalid item code! Try again.
Enter item code and quantity (Press <Enter> key to end): b2 2
Enter item code and quantity (Press <Enter> key to end): <Enter>

 Order number 1
 B1 Chicken briyani X 1 = $5.00
 B2 Mutton briyani X 2 = $11.00
 Total price $16.00
```

The program validates input and checks for the following:

- Item code must be checked before prompting for next order item. If the item code is not valid, display an error message. The item code is in uppercase, but input can be in lowercase.
- Assume that the user will enter a space followed by an integer after the item code. There is no need to validate this.
- When the user presses the `<Enter>` key, indicating no more orders, the program displays the receipt which includes the order number. If the user presses the `Enter` key when the first input line is prompted, it indicates there is no order. In this case, display a message `'No order placed'`.
- The order number is largest number among the keys in the order dictionary plus 1. The first order number starts at 1.

(10 marks)

(b) Update order. This option allows a customer to make changes to the quantity of the order items. The order number remains the same after the update. A sample session is as follows:

```
ABC Briyani
1. Place order
2. Update order
3. Serve order
0. Exit
Enter option: 2
 Order number 1
 B1 Chicken Briyani X 1
 B2 Mutton Briyani X 2

 Order number 2
 B2 Mutton Briyani X 2
 B3 Papadom X 2

Enter order number: 1
 Order number 1:
 B1 Chicken briyani X 1
 B2 Mutton briyani X 2

B1 Chicken Briyani X 1, enter new qty or <Enter> for no change: 0
B1 Chicken Briyani removed.
B2 Mutton Briyani X 2, enter new qty or <Enter> for no change: <Enter>
B2 Mutton Briyani X 2 no change.
Order 1 update:
 Order number 1
 B2 Mutton Briyani X 2

ABC Briyani
1. Place order
2. Update order
3. Serve order
0. Exit
Enter option: 2
 Order number 1
 B2 Mutton Briyani X 2

 Order number 2
 B2 Mutton Briyani X 2
 B3 Papadom X 2

Enter order number: 1
 Order number 1
 B2 Mutton Briyani X 2
B2 Mutton Briyani X 2, enter new qty or <Enter> for no change: 0
B2 Mutton Briyani removed.
Order 1 cancelled!
```

When the update order option is selected, all existing orders are first displayed. If there are no orders, display `'No orders'`. To update an order, the user enters the order number. If the order number is incorrect, display a message `'Order not found!'`. If the order number exists, the order details are displayed. Each item ordered will be displayed, and the new quantity input. If the user presses the <Enter> key, there is no change to the quantity. If the quantity entered is 0, the item is removed. If all the items are removed, the order is considered cancelled. Assume the user will press `<Enter>` key or an integer value for the order number and quantity.

(12 marks)

(c) Serve order. This option allows the stall owner to call out an order number and serve the order. A sample session is as follows:

```
ABC Briyani
1. Place order
2. Update order
3. Serve order
0. Exit
Enter option: 3
Enter order number that is ready: 1
 Order number 1
 B1 Chicken briyani X 2
 B2 Mutton briyani X 2
Press <Enter> when order is collected: <Enter>

ABC Briyani
1. Place order
2. Update order
3. Serve order
0. Exit
Enter option:
```

The user enters the order number. If there are no orders, display `'No orders'`. If the order number exists, the order is displayed. A dummy input with the message '`Press <Enter> when order is collected.`' is displayed. When user presses the `Enter` key, the order menu is displayed. Remove this order from the order dictionary.

(3 marks)

(d) Write the main program that drives this application. Read the food menu from the `menu.txt` file to create a food menu and declare an empty order dictionary before the order menu is displayed.

Include a screenshot of a program execution that shows adding an order, update, and serving an order.

(10 marks)

The question covers concepts in all the seminars. Employ structure programming and use of functions to make the program modular.

Appendix

`menu.txt`

Assume there are no errors in the contents file. Each line consists of the item code, description and price.

```
B1,Chicken Briyani,5
B2,Mutton Briyani,5.5
B3,Fish Briyani,6.0
```

---- END OF ASSIGNMENT ----