COMP609 Assessment Outline and Marking Scheme

	Tasks (the details of these tasks, see the pages that follow)	Marks (total 100)
Task 1 Task 2 Task 3	• (3) Basic console app • (5) Basic GUI app • (7) Database console app	15

Task 4	LMS (Livestock Management System) functionalities:
	• (5) Read data from database and save as objects
	• (5) Calculate livestock statistics
	• (16) Query farm livestock
	• (5) Delete record from database
	• (6) Insert record in database
	• (8) Update record in database
	• Note: all data, whether entered by user or retrieved from database, must be validated. For example,
	  ○ For an entered livestock id, it must be a valid integer.
	  ○ For an entered livestock weight, it must a positive double.
	  ○ For an entered colour, it must be either red, black, or white.
	  ○ For an entered livestock type, it must be either cow, goat, or sheep.
	  ○ For data loaded from database, there could be data rows that are corrupted.
	LMS GUI
	• (32) GUI, including feature design and functional controls
	  ○ (2) Display database records
	  ○ (6) Display livestock statistics and profit forecast
	  ○ (8) Display livestock query results
	  ○ (4) Display deleting database records
	  ○ (6) Display inserting database records
	  ○ (6) Display updating database records
		85

	• (4) Quality programming
	  1. Abstract class and method
	  2. Inheritance
	  3. Polymorphism
	  4. Static class and variables
	  5. Adhere to DRY principle
	  6. Use of self-taught C# features
	  7. Use of self-taught GUI features
	  8. Use of wireframe design	

	• (4) Misc.
	  ○ (2) GUI layout design
	  ○ (2) Testing report demonstrating features of working app	

Assessment Details

Task 1: C# console app – console input and output

Write a program to read two lines of numbers entered by the user in console window, each containing two integers, as shown in the sample run below. The program then calculates all prime numbers between the two pairs, displays the two sets of prime numbers in console window.

Note, your solution must deal with the scenario where the first number may be larger than the second (as shown in the sample run below).

Optional task (1 mark bonus to make up for lost marks): validate user input on the two sets of integers, i.e., each of the two lines must be numerical, contain only two integers, and separated by a comma.

Hint:

1) Split each pair as an array to retrieve the two numbers. Identify each number between the two numbers to see if it's prime.
2) Make use of the IsPrime() method from the lecture.
3) Consider creating a method List<int> GetPrimes(int start, int end). You can reuse this method as the code-behind in Task 2.
4) To convert a list of integers to a single line of strings separated by a delimiter of your choice, you can use `string.Join(<delimiter-string>, <list_name>).`

Sample run:

```
Enter first set of numbers: 21,73
Enter second set of numbers: 40,11
Primes btw 21-73: 23,29,31,37,41,43,47,53,59,61,67,71,73
Primes btw 11-40: 11,13,17,19,23,29,31,37
```

Task 2: GUI app – GUI input and output

Write a GUI app to read two integers entered on GUI by user (use the numbers in Task 1 for testing). Your program then calculates all prime numbers between the two numbers and displays the resulting numbers using GUI controls.

The app must have a **Reset** button that clears up the input and output areas and allows the user to enter again.

Hint, use the functions from Task 1 as the code-behind.

Task 3: DB console app

- Download database file EmployeeData.db from Moodle. Data tables are shown below.

Salesperson table

Id	Fullname	Age	SalesVolume
1001	John Smith	32	$50,000
1002	Lisa Johnson	28	$75,000
1003	Michael Brown	35	$40,000

Manager table

Id	Fullname	Age	TeamSize
1	David Miller	40	10
2	Sarah Davis	38	8
3	Robert Wilson	45	12

Complete the following:

- Create model classes to map the tables in the database. Consider creating a base class containing the shared fields of Salesperson and Manager.
- Read the rows in the tables and save data as objects.
- Display data to console in columnar format.
- Allow object querying via employee id.

Sample run output:

```
 ==Employee List==

 Salesperson   1001     John Smith         32            50000
 Salesperson   1002     Lisa Johnson       28            75000
 Salesperson   1003     Michael Brown      35            40000
 Manager       1        David Miller       40            10
 Manager       2        Sarah Davis        38            8
 Manager       3        Robert Wilson      45            12

Enter employee id: xx
Invalid
Enter employee id: 99
Could not find employee id: 99
Enter employee id: 3
Matching employee:
Manager         3    Robert Wilson        45   12
```

Task 4

Livestock Management System (LMS) App

For this assessment, you are to develop an application using C# and .NET MAUI. A database file FarmData.db is available for download from Moodle. The database contains two tables, Cow and Sheep, as follows.

Cow				
Id	**Cost**	**Weight**	**Colour**	**Milk**
1	46	400	Red	23.5
2	40.5	450.5	Black	27.1
3	43	430	Black	21.9
4	39	330.5	Red	16.4
5	30.5	310	Red	15.8

Sheep				
Id	**Cost**	**Weight**	**Colour**	**Wool**
50001	6.5	22.5	White	9.5
50002	6.5	28.5	White	9.2
50003	6.5	33.2	White	8.9
50004	5.5	23.5	White	8.2
50005	6.4	25.5	Black	9.3
50006	5.4	24.2	White	8.8

Details of Cow and Sheep tables

- Id: a unique identification for a farm animal.
- Cost: in dollars per day, this is the cost for keeping the animal, including water, food, facilities, vet, etc.
- Weight: in kg, the weight of the animal. Government tax is calculated based on weight.
- Colour: the animal colour.
- Milk: in kg per day, the amount of milk produced by cow.
- Wool: in kg per day, the amount of wool produced by sheep.

The following information are provided

- Cow milk selling price: $9.4 per kg.
- Sheep wool selling price: $6.2 per kg.
- Government tax: $0.02 per kg, per day.

To calculate income, cost, and profit

- Income generated by a cow (per day): milk_amount * cow_milk_price

- Income generated by a sheep (per day): wool_amount * sheep_wool_price
- Profit (per day) is calculated as: income – cost – tax

Definition of profit and loss

- Profit: when income is greater than cost, the difference is called profit.
- Loss: when cost is greater than income, the difference is called loss.
- In rare cases where income and cost are the same, the profit is zero.

The application must implement the following compulsory functionalities:

- Read livestock data from database and perform data modelling
  1. Establish database connection.
  2. Create classes to model database tables columns.
  3. Load tables rows as objects in data structures.
  4. Show information about all farm animals by displaying all livestock records. Database records must be displayed in columnar format.
- Livestock statistics reporting and investment profit forecasting
  1. Show how much government tax is paid by the farm per month (30 days).
  2. Show the total profit or loss (depending on the data) of all animals per day.
  3. Show the average weight of all farm animals.
  4. Calculate current daily profit of all cows.
  5. Calculate current daily profit of all sheep.
  6. Based on current livestock average profitability, calculate an estimated daily profit if the farmer were to invest on more livestock. The number and type of livestock (cow or sheep) are specified by the user.
- Query farm livestock. The app should allow the user to query livestock using a combination of livestock type and colour.
  - Possible combinations include:
    - Cows of all colours
    - Black sheep
    - White sheep
    - Black cow
    - Etc.
    - Note that, for type, it's either Cow or Sheep; for colour, it's either all colours, red, black, or white (i.e., you don't have to provide the option of selecting two or more colours).
  - Based on the combination, for the selected livestock, the app should calculate the following:
    1. The total number of selected livestock.
    2. The percentage of the selected livestock.
    3. Profit or loss (depending on the data) per day.
    4. Average weight.
    5. Government tax paid per day.
    6. The produce amount (milk plus wool) per day. If the type is cow, the produce is milk; if the type is sheep, it's wool.
- Database operations
  - **Delete** a row from database table. In case an animal dies, the app should allow farmers to delete the corresponding record from its database table (Cow or Sheep).

- **Insert** a new row into database table. In the likely scenario where there are new arrivals of livestock, the app should allow farmers to add new records. First, ask from the user for the name of the table to insert record into, i.e., Cow or Sheep. Then ask the user to enter all information (weight, colour, etc.) about the new livestock except Id (which will be auto-generated upon inserting). After successfully inserting the record into database, an corresponding object should also be added to the data structure used by the app.
- **Update** existing records in database.
    - The app allows farmers to modify the properties of a livestock. For example, when an animal has gained or lost weight, which would affect the amount of tax to pay.
    - The existing property values should be loaded and presented for the user to make changes.
    - The modification made to livestock objects should be updated in the database and reflected in UI as well.

==Optional Tasks==

The following features are optional. Bonus marks are to make up any lost marks on the compulsory tasks.

- (30 bonus marks) Multi-threading/asyncronous programming and MVVM design pattern.

  Use async/await asynchronous programming to improve user experience and ensure long-running (or possibly failing) operations such as retrieving data from database or web API won't freeze up GUI rendering. And implement the MVVM pattern that decouples the views and business logic by using the commanding interface ICommand.

    - See Microsoft documentation on Data binding and MVVM at: https://learn.microsoft.com/en-us/dotnet/maui/xaml/fundamentals/mvvm.
    - Hint: the CommunityToolkit.mvvm is a great tool for simplifying the implementation.
    - Plenty of other resources on the topic are available by Googling.

==App development suggestions==

- Create wireframes for each view in the app before coding.
- There are plenty of free, easy-to-use wireframing tools, examples include:
    - Figma,
    - Miro,
    - Uizard,
    - Draw.io,
    - Lucidchart.
- Draw a flowchart of all the required functionalities.
- Use the flowchart to divide and allocate which group member takes on which part of the implementation.

==App sample run outputs==

The followings are sample run outputs in the console window for each of the functionalities. However, you are to implement these functionalities using GUI components.

Sample run for reading database records:

```
==Livestock List==
Cow       1         46        400       Red       23.6
Cow       2         40.5      450.5     Black     27.1
Cow       3         43        430       Black     21.9
Cow       4         39        330.5     Red       16.4
Cow       5         30.5      310       Red       15.8
Sheep     50001     6.5       22.5      White     9.5
Sheep     50002     6.5       28.5      White     9.2
Sheep     50003     6.5       33.2      White     8.9
Sheep     50004     5.5       23.5      White     8.2
Sheep     50005     6.4       25.5      Black     9.3
Sheep     50006     5.4       24.2      White     8.8
```

Sample run for livestock statistics:

```
30 days govt tax:                                    $12470.4
Farm daily profit:                                   $667.82
Average weight of all livestocks:                    188.95 kg
```

```
Based on current livestock data:
        On average, a single cow makes daily profit: $84.09
        On average, a single sheep makes daily profit: $44.32
Current daily profit of all sheep is $265.9.
Current daily profit of all cows is $588.62.
```

Sample run for investment forecast:

```
==Estimate Future Investment==
Enter livestock type (Cow/Sheep): cow
Enter investment quantity of Cow: 15
Buying 15 cows would bring in estimated daily profit: $1261.35
```

```
==Estimate Future Investment==
Enter livestock type (Cow/Sheep): sheep
Enter investment quantity of Sheep: 12
Buying 12 sheep would bring in estimated daily profit: $531.84
```

Sample run for querying livestocks:

```
Enter livestock type (Cow/Sheep): xxx
Invalid input
Enter livestock type (Cow/Sheep): cow
Enter livestock colour (All/Black/Red/White): yellow
Invalid input
Enter livestock colour (All/Black/Red/White): all
Number of livestocks (Cow in All colour):   5

Percentage of selected livestock:   45%

Daily tax of selected livestocks:   $384.2

Profit:   $401.92

Average wewight:   384.2kg

Produce amount:   104.8kg
```

```
Enter livestock type (Cow/Sheep): sheep
Enter livestock colour (All/Black/Red/White): red
Number of livestocks (Sheep in Red colour):   0

Percentage of selected livestock:   0%

Daily tax of selected livestocks:   $0

Loss:   $0

Average wewight:   0kg

Produce amount:   0kg
```

```
Enter livestock type (Cow/Sheep): sheep
Enter livestock colour (All/Black/Red/White): white
Number of livestocks (Sheep in White colour):   5

Percentage of selected livestock:   45%

Daily tax of selected livestocks:   $26.38

Profit:   $219.74

Average wewight:   26.38kg

Produce amount:   44.6kg
```

Sample run for inserting a new record in database:

```
====Insert record in database====
Enter livestock type (cow/sheep): xx
Invalid livestock type
Enter livestock type (cow/sheep): cow
Enter cost: 30
Enter weight: 250
Enter livestock colour (black/red/white): yellow
Invalid input
Enter livestock colour (black/red/white): red
Enter milk: 21.5
Record added: Cow       7         30        250       red       21.5
```

```
====Insert record in database====
Enter livestock type (cow/sheep): sheep
Enter cost: 7.5
Enter weight: 30
Enter livestock colour (black/red/white): white
Enter milk: 9
Record added: Cow       8         7.5       30        white     9
```

Sample run for deleting database record:

```
====Delete database record====
Enter livestock id: xx
Invalid input
Enter livestock id: 100
Non-existent livestock id: 100
Enter livestock id: 8
Record deleted: Cow       8         7.5       30        white     9
```

Sample run for updating database record:

```
====Update database record====
Enter livestock id: xxx
Invalid input
Enter livestock id: 100
Non-existent livestock id: 100
Enter livestock id: 7
Enter cost: 40
Enter weight: 240
Enter livestock colour (black/red/white): black
Enter milk: 20.5
Record updated: Cow       7         40        240       black     20.5
```

NOTE: GUI app vs console output

The GUI app will have the exact same functionalities as what's been displayed in the console above, except

- The user input is received via GUI components.
- The output is displayed in GUI components.