Overall project – application development

Weight: 70% of the overall mark.

Introduction

You have a farm which includes several type of livestock. The basic information about the livestock are stored in an access database. Your task is to develop a GUI application that retrieves the data from the database, stores it in efficient data structures, manipulates it and reports useful information related to the profitability of the farm. Details and the expected reports are presented in the sections below.

Learning outcomes: *all (1, 2, 3 & 4)*.

The Access database

The information in the access database is the main input of this program and it provides a clear picture of the different type of livestock and the associated data. The database includes five tables: *Cows, Goats, Sheep, Dogs and Rates and prices.* The fields of each table are described in the subsections below. Notice that the ID of each livestock is unique (two livestock can't have the same ID). A simple database with a few rows for each table is provided to you (find it in moodle). The subsections below describe the tables and the fields of this database. In addition to this, browse the database in moodle to get familiar with the data.

Cows, Goats and Sheep tables

Table 1 describes the fields (columns) of the Cows table in the access database.

Field (column) name	Data type	Comments
ID	Number (integer)	No duplicates
Amount of water	Number (double)	Per day
Daily cost	Number (double)	local currency
Weight	Number (double)	In KG
Age	Number (integer)	In years
Color	String (short text)	None.
Amount of milk	Number (double)	Per day
Is jersy	Yes/No	None

Table 1: Cows table

The table for Goats is the same as the Cows table except it does not include the field 'Is jersy'. The Sheep table differs from the Cow table as follow: it does not include the 'Is jersy' field and instead of the 'amount of milk' field it includes 'Amount of wool' which is also double value but per year and not per day.

Dogs

Table 2 describes the fields for the Dog table.

Field (column) name	Data type	Comments
ID	Number (integer)	No duplicates
Amount of water	Number (double)	Per day
Daily cost	Number (double)	local currency
Weight	Number (double)	In KG
Age	Number (integer)	In years

Color	String (short text)	None.

Table 2: Dog table

Rates and prices

Table 3 describes the fields and rows of the *Rates and prices* table.

Field (column) name: *Rate*	*Amount*	Comments
Goat milk price	Number (double)	Per litre (local currency)
Cow milk price	Number (double)	Per litre (local currency)
Sheep wool price	Number (double)	Per Kg (local currency)
Water price	Number (double)	In Cube (local currency)
Tax	Number (double)	Per KG – per day (not including dogs)
Jersy cow tax	Number (double)	Per KG – per day.

Table 3: rates and prices table

Task and requirements

Your task is to develop an application using C#. Moodle website includes several examples to help you develop the application using C#. If you select to develop the application using Java, talk to your tutor as technical help and support may not be available for you.

Notice that each livestock has a unique ID. One of the main operation that your application must support efficiently is: the user enter an ID and the application display the information associated with this particular livestock. The livestock could be a cow (possibly Jersy cow), dog, sheep or a goat. The most efficient data structure for this operation is the hash table. If you do not use hash table, you will lose marks as detailed in the marking criteria. Notice that there are several versions of hash table (also Dictionary is a version of hash table) and you are allowed to use any of them. Also you are allowed to use all the built-in data structures provided by your development environment (Visual studio).

As mentioned above, in moodle you will find a simple access database with a few rows. It is your responsibility to build or obtain (from other students) a more comprehensive database. Also you should manually test (basic testing) all components of your application.

Functionality

The application should have an appropriate Graphical user interface. The implementation should be able to connect and retrieve data from the database. While reading rows from tables in the database, the implementation should create objects associated with these rows (livestock). All objects should be stored in a hash table where the ID is the key and a pointer to the object is the value. After this the connection to the database can be closed. Now the application is ready to create reports based on user inputs. Table 4 includes details of all reports the implementation should have (the GUI should enable the user to enter a number and the application displays the information related to the report number as detailed in Table 4).

Also, it is expected that your implementation incorporates inheritance and polymorphism.

Report number	**Description**	Comments
1	The user enter an ID and the program displays the information associated with this animal farm. In addition to the basic information, a string will be added to state the type of the animal (Dog, Cow, Jersey Cow, Sheep or Goat)	7 marks
2	Display the total profitability/loose of the farm per day	5 marks
3	Display the total tax paid to the government per month	5 marks
4	Display the total amount of milk per day for goats and cows	5 marks
5	Display the average age of all animal farms (dog excluded)	4 marks
6	Display the average profitability of “Goats and Cow” Vs. Sheep	5 marks
7	Display the ratio of Dogs’ cost compared to the total cost	5 marks
8	Generate a file that contains the ID of all animal ordered by their profitability (You are not allowed to use built-in sorting algorithm – Your code must do the sorting). Dogs are excluded.	7 marks
9	Display the ratio of livestock with the color red	4 marks
10	Display the total tax paid for Jersey Cows	5 marks
11	The user enter a threshold (number of years), and the program displays the ratio of the number of animal farm where the age is above this threshold.	4 marks
12	Display the total profitability of all Jersey Cows.	4 marks

Table 4: list of the reports the application should have.

Tips and implementation guidelines

This project encourages you to make reasonable choices. There is no ‘one way’ to implement this solution. Furthermore, it is important that you have a long-term plan to develop and test components step by step. In the following, one approach to develop this application is presented which includes tips and guidelines.

1. The best start is to understand the requirements and the “big picture”. I will read this document a few times and identify the components and the required skills to implement this application. It is a good idea to have a high-level design and a plan to implement this project.
2. Get familiar with the basic features of GUI in (C#). Moodle includes activities and resources to help you gain the necessary skills.
3. It is important to know how to connect and retrieve data from tables in access databases. Go to moodle and do the relevant activities which are very close to what you need for this project. Pay attention that these examples show you how to retrieve the information row by row from all tables.
4. In this step focus on the Goats table only. Write a Goat class with properties that match the information. Read the Goats table row by row and for each row, create an object. Test your code by printing to the Console screen or a textbox.
5. This step is a follow up of Step 3. Declare a hash table (see moodle for examples on hash table). For each object, insert it to the hash table where the ID is the key and the object is the value (a pointer to the object). For testing purposes, from GUI, search using the ID and expect my code to show the correct information associated with the specific goat. You can add a method in the Goat class which displays the associated information.
6. Now you have to think about the entire application and the way you want to have your classes. Remember that you have to demonstrate inheritance and polymorphism. I would have a basic class called livestock which includes all the functionality common to all livestock. After this, I

will have Goat class, Cow class, Sheep class, Dog Class and ‘Cow New Jersy’ where some of them inherit from each other and all of them inherent from livestock.

7. It is a good idea to complete the code by looking at the reports listed in Table 4.

8. At this step it is recommended that you polish the code, more comments, more testing, removing unnecessary code and add more improvements.

Marking criteria

Please pay attention to the marking criteria in Table 5.

	Description	Mark
1.	Reports as listed in Table 4	60
2.	Demonstration of inheritance	2
3.	Demonstration of polymorphism	3
4.	Error handling	5
5.	Comments and indentation	3
6.	Short methods	2
7.	Good object-oriented practice	5
8.	Efficient algorithms(use of hash table and sorting algorithm)	10
9.	Appropriate number of classes	5
10.	Evidence of testing all functions of the program: Show screenshots.	3
11.	Self-marking	2

Table 5: marking criteria