## Programming Concepts II (PROG1815): In-Class Supplemental Part A – Pet Record Management

In this class you will practice defining and using classes to manage record systems with the use of a listbox. Specifically, you will: define a class, instantiate an object of that class, and make use of its properties and method(s).

1. Create a class called ***Pet:***
   1. Define 8 fields:
      1. *RegistrationNumber*
      2. *Name*
      3. *StreetAddress*
      4. *City*
      5. *PostalCode*
      6. *Province*
      7. *PetType*
      8. *PetDateOfBirth*
   2. Create a default constructor.
   3. Use the **Get** and **Set** accessors to access the fields.
2. Create a form that accepts the fields defined in 1. above. Create a **list<Pet>** defined in 1. above. You are required to do the following:
   1. Create a **list<Pet>** defined in 1. Above.
   2. Add textboxes and labels to capture and display the fields defined in 1 above.
   3. Add a **Save** button to add / update pet objects to the list. Validation – all fields are required, RegistrationNumber should be numeric, PetDateOfBirth is of type Datetime. Duplicate registration numbers are not allowed. If the *RegistrationNumber* exists, **update** the pet records. Otherwise, **add** a new pet record.
   4. Add a **Delete** button to remove pet objects from the list. Search should be done by *RegistrationNumber*.
   5. Add a **Find** button, label and a textbox to search the list by registration number. Once found, populate the fields with data from the list.
   6. Add a **listbox** to display data from the list. The Pet name should be displayed and the registration number will be the value member.
   7. On selecting an item from the **listbox**, use the SelectedIndexChanged event handler to populate the fields on the form.
   8. A screenprint of the expected form is shown on the next page.
3. NOTE: Users must be able to add multiple records to the list. Duplicate registration numbers are not allowed. The listbox should be re-populated after all edits are complete. Clear the text fields after the Add, Update and Delete buttons are clicked and updates are done successfully.

## Marking:

* (3) The solution works correctly and follows the specified approach, e.g. the class is defined, instantiated and the properties and method are used to work with the list and listbox.
* (2) The solution works correctly but did not follow the specified approach.
* (1) The solution does not work correctly but an attempt was made at following the specified approach.
* (0) Either no submission or an effectively empty solution is submitted.

## Programming Concepts II (PROG1815): In-Class Supplemental Part B – Exam Review

In this class you will practice defining and using classes to manage record systems with the use of a listbox. Specifically, you will: define a class, instantiate an object of that class, and make use of its properties and method(s).

For the Pet class provided in Part A above, you are required to do the following:

1. Override the ToString method in the Pet class to create a tab delimited string of records from Part A above.
2. Create an instance method called *SaveRecordToFile* to append or update a tab delimited record to a file named **file1.txt** stored in the default location. Validation – all fields are required, RegistrationNumber should be numeric, PetDateOfBirth is of type Datetime. Duplicate registration numbers are not allowed. If the *RegistrationNumber* exists, **update** the pet records. Otherwise, **add** a new pet record.
3. When the **Save** button in Part A is clicked, call the method *SaveRecordToFile*
4. Create an instance method called *DeleteRecordFromFile* to remove pet objects from the file. Search should be done by *RegistrationNumber*.
5. A screenprint of the expected form is shown on the next page.

## Marking:

* (3) The solution works correctly and follows the specified approach, e.g. the class is defined, instantiated and the properties and method are used to work with the list and listbox.
* (2) The solution works correctly but did not follow the specified approach.
* (1) The solution does not work correctly but an attempt was made at following the specified approach.
* (0) Either no submission or an effectively empty solution is submitted.

