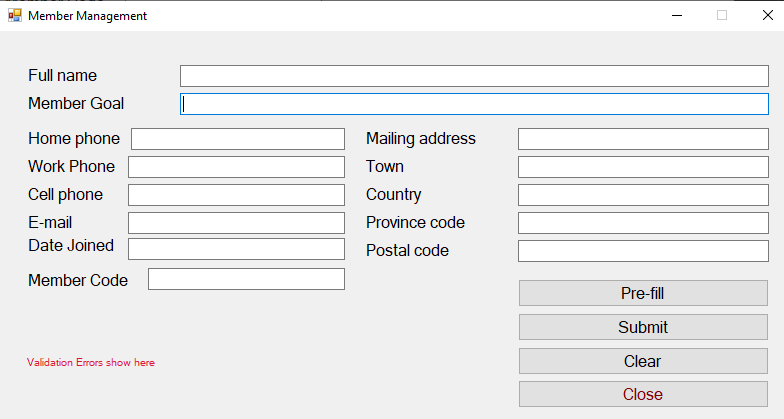
# Programming Concepts II – Assignment 2 – String Manipulation

This assignment is an exercise in using string manipulation to edit and reformat input fields. Some fields have patterns, some expect numeric values … possibly with limits~~. Some fields are required (and cannot be just spaces), some are optional or conditionally optional, some will cause abends if null is not expected. Strings need to be trimmed of leading and trailing blanks, shifted to upper- or lower-case or capitalized.~~

This is also an exercise in creating and using reusable code … code usable by multiple projects.

**~~Wherever you see “XX”, replace it with your initials~~**

1. ~~Create a~~ ***~~XX~~****~~Validation~~* ~~class with the following~~ ***~~static~~*** ~~methods:~~
   1. **~~XX~~**~~Capitalize – this takes a string parameter and returns a string. Return an empty string if the input is null or empty. It should trim leading and trailing spaces, and change the first letter of each word (if it is a letter) to upper case and the remaining letters in each word to lower case. Watch 1-letter words.~~
   2. **~~XX~~**~~MemberCodeValidation - This takes a string and returns a Boolean. If the incoming string is null or an empty string, pass it (so method can be used where this is optional). Otherwise, return true if the string is 9 characters long and contains exactly 5 numbers and 4 upper-case letters in any order: “AAAA66666” & “5A5A5AA55” are both valid.~~
   3. ***~~XX~~****~~PhoneNumberValidation~~* ~~– this takes a string and returns a Boolean. If the incoming string is null or empty, pass it. Otherwise, return true only if the entire string fits the phone pattern “123 123 1234” or “123.123.1234”. Note for the space separator or period separator, either one can be used but not both (e.g. 123 123.1234 would be incorrect).~~
   4. **~~XX~~**~~UKPostalValidation – this takes a string by reference and returns a Boolean~~. ~~If the incoming string is null or empty, pass it. Otherwise, return true only if the string’s entire content fits the UK postal pattern “A[A]3[A/3] 3AA”~~ (~~where “3” represents any digit, “A” represents any letter, [A] represents an optional letter, and [A/3] represents an optional letter or digit). Accept upper or lower case,~~ *~~with or without the space~~*. If valid, shift the input string to uppercase characters and insert the space, **if missing.**
2. Create the following form, naming it ***XXGymMemberManagement.cs*** 
3. Buttons:
   1. ~~Pre-Fill” loads the form with valid data … making it easier to test individual field errors.~~
      1. ~~Use this string as the Pre-Fill value for the “Member Goal” field:~~ **~~“To Run 126 mileS; in 2 hours. Perform 300 situps, and Squat 100 pOUNDS, by January of 2021.” -> reform this to be valid~~**
   2. ~~“Submit” edits the current data on the form and reformats the fields back onto the form.~~
      1. ~~All validation errors are displayed together on the form in a red label, rich text box or in a message box … one line per error, with focus to the first field in error.~~
   3. ~~“Close” closes the form.~~
4. ~~Tab sequence should be down the top rows and the left column, then down the right column and the buttons.~~
5. ~~Perform the following validations (all fields are optional unless specified otherwise). All error messages are to be shown at once, one line per error, focus moved to the first field in error.~~
   1. *~~Full name~~* ~~is required and must contain more than one name (eg First Name and Last Name).~~
   2. ~~At least one phone number is required. Use your~~ ***~~XX~~****~~PhoneNumberValidation~~* ~~method to verify the phone number(s) provided. -> if it is provided it should be more than one~~
   3. ~~A non-blank~~ *~~Member Goal~~***~~is required~~**~~.~~
   4. *~~Email~~* ~~must be a valid email pattern, if provided. There’s a MailAddress class in the System.Net.Mail namespace that throws an exception when~~ *~~instantiated~~* ~~with an invalid email address. There’s also a plethora of indecipherable regular expressions out there.~~
   5. ~~A non-blank~~ *~~Member Code~~***~~is required~~**~~. Use~~ **~~XX~~**~~MemberCodeValidation to validate the code provided. -> optional~~
   6. ~~The~~ *~~Date Joined~~* ~~cannot be in the future and should display in the format 2021 Jan 15.~~
   7. *~~Mailing address~~* ~~and~~ *~~Town~~*~~, if provided, must contain at least 3 letters.~~
   8. *~~Province Code~~*~~, if provided, must be exactly two~~ **~~letters~~**~~, not just two characters.~~
   9. *~~Postal code~~***~~is required~~**~~. Use~~ **~~XX~~**~~UKPostalValidation to validate and format it.~~
   10. ~~If~~ *~~Email~~* ~~is not provided, then~~ *~~Mailing Address~~*~~,~~ *~~Town~~* ~~and~~ *~~Province Code~~* ~~are required.~~
6. Perform the following data formatting
   1. ~~Use your~~ **~~XX~~**~~Capitalize method to format Full name, Member Goal, Mailing Address and Town.~~
   2. ~~Remove all punctuation characters from Member Goals except the period(“.”).~~
   3. ~~Shift the province code to upper case.~~
   4. ~~Shift the email to lower case (it’s still valid, but easier to use for comparisons~~).
   5. ~~The Postal code should be reformatted by~~ **~~XX~~**~~UKPostalValidation~~.

## Hand In

1. Zip and upload your project folder to the assignment 2 drop-box for this course.