Question 2

The objective of the application is to provide a method by which a society can manage member details, storing their full name, associated member ID, and whether the membership is active or not. It is assumed that a member may never rejoin the society, so functionality only includes the deactivation of a membership, without the reactivation.

The application itself is based on the example '*Quicksort*' program that was provided for usage during the AC52002 module. Only minor changes were needed to enable it to cooperate with searching member class attributes.

The basis by which is performs the sort is by concatenating all details into an identifier string, allowing these to be presented to the quicksort algorithm so that the sort procedure only has to be performed once (note: the sort is currently set to run after importing the initial test set & also after each iteration of the switch cycle. While this could easily be incorporated to be called only after adding a new member, keeping the sort call within the main allows for minimal changes to be made in the code to detect if a sort is performed correctly (i.e. remove the break;(line 51) from the first option & display is performed immediately afterwards.) For the identifier string, it takes on the format of YYYYMMDD<Surname><Unified\_whitespace><Forename>, with the unified whitespace being based on the assumption that the longest surname is <700 characters (https://en.wikipedia.org/wiki/Hubert\_Blaine\_Wolfeschlegelsteinhausenbergerdorff,\_Sr.).

The application is currently set to allow for 10 members (line 25), this was done for performance reasons during testing, but can be increased to the 10,000 amount if desired. If this is done, I also suggest moving the sort calls (lines 60-62) into the case statement for NewMember (Line 50), to be performed after the new member insertion. This should be done, as a long sort after each option is pressed is not a desirable outcome if the user has not inserted any new records. Included in the application is a database file with 10 records

Inputs are validated using several validation functions - these include one for dates, one for names, and one for numerical input. The validation function for numerical input has a switch to allow for a single character (for the menu) and multiple characters (for all other situations). The date validation does not make use of regular expressions, as it needs to be pulled apart into separate variables anyway, as it is later plugged into an algorithm to ensure the date entered is legitimate. The algorithm used is a less elegant form of Zellers congruence, as only the day range is needed, not day itself. A caveat has to be put in place to check for Februarys day range, as the algorithm does not function for this month - this is fine, as it must be checked for the leap year anyway.

Validation implemented is positive-only, meaning that validation only passes if the application explicitly detects a correct input, and is not assumed. In the tables overleaf is a sample of inputs and their related results. Results of the legacy member search are not included, but work as intended.

Menu:

|  |  |  |
| --- | --- | --- |
| Entered value | Expected Result | Actual Result |
| <blank> | Invalid | Invalid |
| 0 | Valid | Valid |
| 1 | Valid | Valid |
| -1 | Invalid | Invalid |
| A | Invalid | Invalid |
| 4 | Invalid (Pass input validation, but invalid menu option) | Invalid (Pass input validation, but invalid menu option) |
| 1A | Invalid | Invalid |

Name input:

|  |  |  |
| --- | --- | --- |
| Entered Value | Expected Result | Actual Result |
| <blank> | Invalid | Invalid |
| 1 | Invalid | Invalid |
| 123 | Invalid | Invalid |
| A | Valid | Valid |
| abc | Valid | Valid |

Date input:

|  |  |  |
| --- | --- | --- |
| Entered Value | Expected Result | Actual Result |
| <blank> | Invalid | Invalid |
| 1 | Invalid | Invalid |
| 0 | Invalid | Invalid |
| 111 | Invalid | Invalid |
| 1111111111 (length=true) | Invalid | Invalid |
| 11-11-2000 | Invalid | Invalid |
| 30/02/2000 | Invalid | Invalid |
| 29/02/2000 | Valid | Valid |
| 29/02/2001 | Invalid | Invalid |
| 28/02/2001 | Valid | Valid |
| 31/04/2000 | Invalid | Invalid |
| 30/04/2000 | Valid | Valid |
| 31/01/2000 | Valid | Valid |