PROG1165 – Advanced Software Techniques

Assignment 4 – Input Validation

This assignment is to be done in groups of 2 or 3 or 4. It is your responsibility to make or find a group (there is a discussion topic available for this).

Note: This assignment is worth three times as much as other assignments.

There are three parts to this assignment, with three separate dropboxes:

1. Your team must examine the requirements for the program and come up with a *requirements list* for input data validation for the program. This is handed in at the first deadline. A master requirements list for input data validation will be generated from all teams' submissions that you will use when writing the program. It will be posted on D2L by the end of the day after the 1st deadline. It is your responsibility to check D2L for this.
2. Your team will write the program, based on the description below and the additional master requirements list for input data validation. It must be handed in by the second deadline. You cannot hand it in more than 8 hours late (this is a major deviation from the normal late policy). A ZIP file with all submitted executables will be created. It will be posted on D2L on the day after the executables are due. It is your responsibility to check D2L for this.
3. Your team will test the executables for bugs. Your bug list must be handed in by the third deadline. **PLEASE CHECK THE NEWS ITEMS IN D2L FOR THE LIST OF EXECUTABLES YOU WILL BE RESPONSIBLE FOR TESTING.**

The Requirements List (Milestone 1)

Your requirements list for input validation should not be a formal document.

It *must* be a Word document called *req.doc* or *req.docx*. It must have the first and last names of each group member at the top.

The requirements should be a bullet list oriented towards being friendly for the user while still maintaining the integrity of the input data.

Your unique identifier will be provided as part of the feedback to your submission in D2L.

The Program (Milestone 2)

Create a program that takes input for an address book (of people, not businesses), validates the input, and stores the data in an array. Once the user has entered the desired number of entries, the address book should be displayed in its entirety.

The data stored consists of:

- Name (maximum length 30 characters)
- Street Address (maximum length 60 characters)
- City (maximum length 60 characters)
- Province (in the format of a 2 character long Canadian province abbreviation used by Canada Post)
- Postal Code (in the format of a Canadian Postal Code)
- Phone Number, including area code

As noted earlier, the specific requirements for input validation will be provided after the first deadline. It is a compilation of all submitted requirements.

You must prompt the user in a user-friendly way. This requires that the prompt be informative but not overbearing.

If the user enters invalid input, an error message must be displayed indicating what the problem is. The user must be given an option as to whether to skip entering this part of the address book entry (e.g. skip the Address and continue with the City) or to try again.

The array of addresses is of size 10. The user will be permitted to enter up to 10 entries. If the user chooses to end entry early, they can specify this by pressing ENTER as the first and only character of the name (this is the lone exception to the "pressing ENTER skips a component" requirement).

Once the user has done 10 entries or has chosen to end entry early, the program will then display the contents of the address book, three entries on a page. Each entry will be five lines in size. The lines must be in the following format:

Fred Flintstone
11 Cobblestone Lane
Bedrock, ON, N2Y 3E8
519-555-1212
++++++++++++

Paginate your output properly (the user should not have to scroll to see output).

Do not use regular expressions or any third-party libraries for this assignment.

When the program starts up, display the string that your group will be provided. This will uniquely identify your assignment. Do not put **any other identifying information** (e.g. "Fred's Address Book) in the program.

Call your executable file **astA4.exe**.

Submit your solutions and your executable to the appropriate D2L dropbox. *If you do not have an executable submitted for this assignment, you will lose 40% of the marks assigned to milestone 2.*

The Testing (Milestone 3)

A ZIP file containing all executables will be put on D2L. Download this ZIP file. Test each program except for your own, looking for bugs (not only in input validation but in all aspects of the program). Document each bug that you find in others' programs. The documented bugs must be in a file called bugs.pdf and is of the following format:

- each program's test results must start on a new page
- each page's header must have the "uniquely-identifying string" displayed upon startup of the particular tested program (google "Word 2010 sections header" if you don't know how to make different headers in a Word document)
    - put the pages in alphabetical order, sorted by the uniquely-identifying string
- each page must have the bugs found listed in a table with the following columns:
    - Type of Bug (e.g. Input, Output, Requirement, Other)
    - Description
    - How to Duplicate

PLEASE CHECK THE NEWS ITEMS IN D2L FOR THE LIST OF EXECUTABLES YOU WILL BE RESPONSIBLE FOR TESTING.

Evaluation

Your mark will be based as follows:

- 50%: Instructor Evaluation
- 25%: How effective you are at finding others' bugs
- 25%: How few bugs others found in your program

You are competing with other teams on the last two 25% items. Those who find the most or best bugs will get the highest mark. Likewise, those who have the fewest bugs in their own programs will have the highest mark. The instructor’s evaluation will be used as a baseline in the case that few bugs are found by others. If the instructor’s evaluation doesn't find many bugs, then we can all be happy.

This assignment has some very strict submission requirements. This is so that we can have the collation and sharing of information go as smoothly as possible. Penalties will be significant for late submissions.