Input Validation

The goal of this assignment is to produce a program that validates its input using regular expressions.

This will be an individual assignment (no teams).

Detail:

Produce a command-line driven telephone listing program. The program shall be capable of receiving and storing a list of people with their full name and telephone. The program shall include the following commands:

- ADD "<Person>" "<Telephone #>" - Add a new person to the database
- DEL "<Person>" - Remove someone from the database by name
- DEL "<Telephone #>" - Remove someone by telephone #
- LIST - Produce a list of the members of the database

The above shall all be specifiable on the command line directly (e.g. <executable name> <mode> <arguments>) so that testing and verification of functionality can be scripted. When no arguments are specified, a usage (help) screen shall be displayed.

Create regular expressions for <Person> and <Telephone #>. Use these regular expressions to verify that the user is supplying valid data. More flexible specifications will be graded higher. For example:

- Allowing for international or US format telephone numbers
- Allowing for <first middle last>, <first last> or <last, first MI>)

Reject any attempts to provide invalid data. When valid input is provided, an exit code of 0 shall be returned; when invalid input is provided, an exit code of 1 shall be returned and an appropriate error message displayed to standard error (STDERR). An attempt to remove a non-existent name from the directory shall return an exit code of 1.

You have the freedom to choose the implementation that you are comfortable with for persisting the phonebook to disk (e.g. XML, text file, binary file, CSV, database, etc.). If you are attempting the bonus at the bottom of the assignment, you might choose to use a SQL database initially to avoid rework later.

Permissible languages: C/C++, Any .Net Language, Java, Perl, Python, others with permission

Sample Inputs:

Acceptable inputs for name:

- `Bruce Schneier`

- `Schneier, Bruce`
- `Schneier, Bruce Wayne`
- `O’Malley, John F.`
- `John O’Malley-Smith`
- `Cher`

Unacceptable inputs for name:
- `Ron O’’Henry`
- `Ron O’Henry-Smith-Barnes`
- `L33t Hacker`
- `<Script>alert(“XSS”)</Script>`
- `Brad Everett Samuel Smith`
- `select * from users;`

*Acceptable inputs for phone: *remember these are also international numbers*
- `12345`
- `(703)111-2121`
- `123-1234`
- `+1(703)111-2121`
- `+32 (21) 212-2324`
- `1(703)123-1234`
- `011 701 111 1234`
- `12345.12345`
- `011 1 703 111 1234`

Unacceptable inputs for phone:
- `123`
- `1/703/123/1234`
- `Nr 102-123-1234`
- `<script>alert(“XSS”)</script>`
- `7031111234`
- `+1234 (201) 123-1234`
- `(001) 123-1234`
- `+01 (703) 123-1234`
- `(703) 123-1234 ext 204`

The TA will utilize a script to test your program using all of the above acceptable and unacceptable inputs for name and phone number, as well as additional ones not listed here.

Submission instructions:

You are required to provide the source code including instructions on how to compile and run the software. Please indicate if your code has any dependencies (e.g. libraries, external programs, etc.)

that must be installed prior to running your code. I would strongly prefer that you do not rely upon anything that is not freely available. If this is not possible, please let me or the TA know in advance so we can work out some way for us to evaluate and grade your assignment. In addition to the code and instructions, you must also turn in a report that describes your submission. This report should include a description of how your code works, the design of your regular expressions, any assumptions you have made, and the pros/cons of your approach.

Bonus (10 points):

Assuming a database backend wasn't used, change to use a SQL database engine, such as SQLite, to persist the phonebook to disk. For reading and writing to the database, use an API that supports parameterized queries (also known as prepared statements) in SELECT and INSERT statements which will avoid SQL insertion vulnerabilities.

Grading Criteria:

- The source code should be provided along with submission ({**0 point out of 100 if there is no code attached**})
- Instruction of how to build (if applicable) and execute the code (**(-20) points if there is no instruction**)
- Report **<<40 points>>:**
    - Description of how your code works (10 points)
    - Compilation instructions (10 points)
    - Assumptions you have made (10 points)
    - Pros/Cons of your approach (10 points)
- Program is running successfully **<<60 point>>**:
    - ADD operation (20 points)
    - DEL operation (20 points)
    - LIST operation (20 points)
- Bonus <<**10 points**>>:
    - Using database to store the input data (5 points)
    - Using an API that supports parameterized queries (prepared statements) (5 points)