Course: CSC 340.04

Student: Kullathon "Mos" Sitthisarnwattanachai, **SFSU ID:** 921425216

Teammate: n/a, **SFSU ID:** n/a

Assignment Number: 03

Assignment Due Date & Time: 03-09-2021 at 11:55 PM

Important

This project was built and tested using CMake 3.21 and C++20. For instructions on how to build and run each part, please **refer to the README file** in the submission folder.

All parts have been tested to produce identical output. Screenshots are provided as a sample for each part.

PART A – TIC TAC TOE, 5 points

Please implement a basic version of Tic Tac Toe:

1. Function **main** and function headers are provided. Please implement the functions and

   do not change the **main**.

   - See part_a folder.

2. Our program must produce **identical** output:

   Assignment-03_PA_Run1.txt and **Assignment-03_PA_Run2.txt**

   - See below for screenshots demonstrating the program.

```
-------------
|   |   |   |
-------------
|   |   |   |
-------------
Enter a row (0, 1, 2) for player X   : 0
Enter a column (0, 1, 2) for player X: 0


-------------
| X |   |   |
-------------
|   |   |   |
-------------
|   |   |   |
-------------
Enter a row (0, 1, 2) for player O   : 0
Enter a column (0, 1, 2) for player O: 0
This cell is already occupied. Try a different cell
Enter a row (0, 1, 2) for player O   :
```

```
-------------
| X | X |   |
-------------
| O |   | O |
-------------
Enter a row (0, 1, 2) for player X   : 1
Enter a column (0, 1, 2) for player X: 2


-------------
| X | O |   |
-------------
| X | X | X |
-------------
| O |   | O |
-------------
X player won


Process finished with exit code 0
```

```
-------------
|   |   |   |
-------------
|   |   |   |
-------------
|   |   |   |
-------------
Enter a row (0, 1, 2) for player X   : 1
Enter a column (0, 1, 2) for player X: 1

-------------
|   |   |   |
-------------
|   | X |   |
-------------
|   |   |   |
-------------
Enter a row (0, 1, 2) for player O   : 0
Enter a column (0, 1, 2) for player O: 2
```

```
-------------
| X | X | O |
-------------
| X | O | X |
-------------
Enter a row (0, 1, 2) for player X   : 0
Enter a column (0, 1, 2) for player X: 1

-------------
| O | X | O |
-------------
| X | X | O |
-------------
| X | O | X |
-------------
No winner

Process finished with exit code 0
```

PART B – Credit Card Number Validation, 5 points

Credit card numbers follow certain patterns. A credit card number must have between 13 and 16 digits. The starting numbers are: 4 for Visa cards, 5 for MasterCard cards, 37 for American Express cards, and 6 for Discover cards.

Example: Validating 4388576018402626

   a) Double every second digit from right to left. If doubling of a digit results in a two-digit number, add the two digits to get a single digit number.

   b) Now add all single-digit numbers from **Step a**:

      $4 + 4 + 8 + 2 + 3 + 1 + 7 + 8 = 37$

   c) Add all digits in the odd places from right to left in the card number:

      $6 + 6 + 0 + 8 + 0 + 7 + 8 + 3 = 38$

   d) Sum the results from **Step b** and **Step c**:
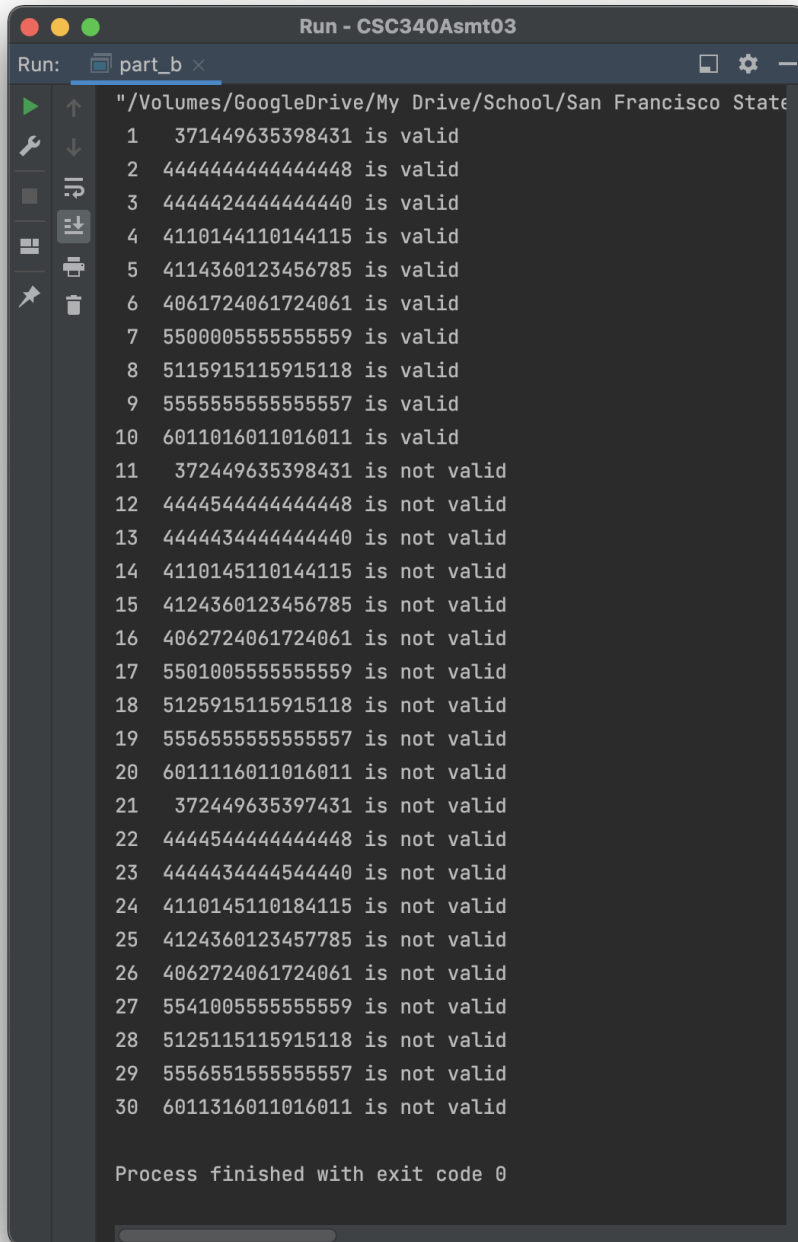
      $37 + 38 = 75$

   e) If the result from **Step d** is divisible by 10, the card number is valid; otherwise, it is invalid.

Please implement Credit Card Number Validation:

   1. Function **main** is provided. Please implement **isvalidcc** and other functions which you may add to the program.

      ● See part_b folder.

2. Please do not change function **main**

3. Your program must produce **identical** output: **Assignment-03_PB_Run.pdf**

- See below for a screenshot demonstrating the program.

```
                    Run - CSC340Asmt03
Run:    part_b ×                                     ⬜ ✿ ─
      "/Volumes/GoogleDrive/My Drive/School/San Francisco State
    1     371449635398431 is valid
    2    4444444444444448 is valid
    3    4444424444444440 is valid
    4    4110144110144115 is valid
    5    4114360123456785 is valid
    6    4061724061724061 is valid
    7    5500005555555559 is valid
    8    5115915115915118 is valid
    9    5555555555555557 is valid
   10    6011016011016011 is valid
   11     372449635398431 is not valid
   12    4444544444444448 is not valid
   13    4444434444444440 is not valid
   14    4110145110144115 is not valid
   15    4124360123456785 is not valid
   16    4062724061724061 is not valid
   17    5501005555555559 is not valid
   18    5125915115915118 is not valid
   19    5556555555555557 is not valid
   20    6011116011016011 is not valid
   21     372449635397431 is not valid
   22    4444544444444448 is not valid
   23    4444434444544440 is not valid
   24    4110145110184115 is not valid
   25    4124360123457785 is not valid
   26    4062724061724061 is not valid
   27    5541005555555559 is not valid
   28    5125115115915118 is not valid
   29    5556551555555557 is not valid
   30    6011316011016011 is not valid

   Process finished with exit code 0
```

PART C – Dictionary 340 C++, 90 points

Our satisfied clients are back to ask us to implement another interactive dictionary. Our

dictionary takes input from users and uses the input as search key to look up values associated

with the key. Requirements:

- **Coding**: No hard coding, https://en.wikipedia.org/wiki/Hard_coding. *Please think about*

  Dynamic and Scalable.

- **Data Source**: A text file, **Data.CS.SFSU.txt** . *Please think about Software Deployment*

  and Usability.

- **Data Structure**: Use existing data structure(s) or create new data structure(s) to store

  our dictionary's data. Each keyword, each part of speech, and each definition must be

  stored in a separate data field. Do not combine them such as storing three parts in one

  String.

- **Data Loading**: When our program starts, it loads all the original data from the Data

  Source into our dictionary's data structure. The data source file is opened once and

  closed once per run. It must be closed as soon as possible. It must be closed before

  our program starts interacting with users.

- **User Interface**: A program interface allows users to input search keys. This interface

  then displays returned results. Our program searches the dictionary's data (not the data

  source text file) for values associated with the search keys.

- **Identical Output**: Our program's output must be identical to the complete sample run's

  output: **Assignment-03_PC_Run.pdf**

1. **Program Analysis to Program Design, 10 points.** *Please think about Interviews.*

In at least 1 full page, please <u>focus on the differences/improvements you are making in</u> <u>this C++ version of the program in comparison to your previous Java version</u> while explaining the following **in detail**:

- Your analysis of the provided information and the provided complete sample output. *Please think about Clients and Sales.*

    - The complete sample output for part C in this assignment are largely identical to the one provided for assignment 02. The key difference in terms of the program's functionality in this assignment is the ability to load the definition from an external source.

        The sample output demonstrates the data loading component, where the program attempts to open a file from an absolute path and fails. In this implementation, the same absolute path is hard-coded as the default path in order to produce an identical output as the sample. The user can provide a path relative to the current working directory in order to produce the same output.

- What problem you are solving. Please explain it clearly then define it concisely. *Please think about Problem Solving and Interviews.*

    - As mentioned, the key difference between this part of the assignment and the previous version is the way we handle the source data.

In the previous assignment, the way in which we structure the enum provides us with a degree of certainty of how the data is meant to be stored and parsed. In addition, the previous assignment required the students to extrapolate the data from the sample output. Then, store and utilize them in such a way that would produce an identical output.

In this part of the assignment however, a text file was provided as a data source in a particular format, and students are required to parse the given data to produce a certain output.

- How you store data from the external text file. And why. *Please think about Data Structures and Data Design.*

    - The dictionary data is provided in a text file. Since students aren't allowed to modify the source file, we don't really have much control over how it is stored, but rather how we interpret the file.

    Storing the definitions in the text file allows for the dictionary to be expanded later on without the need to edit and recompile the source code. This model is much more scalable than the approach used in the last assignment, where we store the definitions in an enum class. Storing the definitions in an enum class can still be considered hard-coding, as it requires changes to the code i.e., the enum class, directly in order to expand the dictionary.

    Upon inspection of the data source file, we see that each line holds a unique term. And each term has one or more definitions, whereby each

definition is delimited by a vertical pipe character ("|"). In addition, each

definition has an associated part of speech, which is delimited by an

arrow (" -=>> ") surrounded by spaces.

We start by reading each line, then splitting it at the vertical pipe. The

first item in each line must be the term, and each item after that are the

definitions. For each definition, finding and splitting at the first arrow

allows us to extract the part of speech and the definition. They are then

loaded on to different data structures, which we define below.

- Which data structures you use/create for your dictionary. And why. *Please think about Data Structures and Data Design.*

  - Like the Java version, a simple struct is used to hold each definition.

    Each DictEntry struct contains the definition and the part of speech it

    applies to. The struct also contains overloaded operators for comparison,

    which allows them to be sorted and checked for duplicates using STL

    functions.

    The terms are placed in an unordered_map, where the key is the term

    itself, and the value is a vector containing DictEntry objects.

    In addition, QueryArg struct is used to hold valid query arguments and

    the valid tokens that may be used to invoke them. In the previous

    assignment, enums were used for this purpose. But since enums may

    only represent integers in C++, we limit their use for identifying certain

    argument types (as a member of QueryArg). Together, they are used to

validate the given search options and to display appropriate error

messages.

2. **Program Implementation, 80 points.** *Please think about Interviews.*

- Implement your program to meet all the requirements.

  - See the part_c folder.

    In order to be able to provide the relative path to the data source at the

    start of the program in the manner specified by the sample output,

    ensure that the current working directory is the location where the data

    source is located.

    Refer the README for more information.

- In your assignment report, demonstrate your program to your grader/client.

  - See below for screenshots demonstrating the program.

```
! Opening data file... C:\Users\MickeyMouse\AbsolutePath\DB\Data.CS.SFSU.txt
<!>ERROR<!> ===> File could not be opened.
<!>ERROR<!> ===> Provided file path: C:\Users\MickeyMouse\AbsolutePath\DB\Data.CS.SFSU.txt
<!>Enter the CORRECT data file path: ./Data.CS.SFSU.txt
! Loading data...
! Loading completed...
! Closing data file... ./Data.CS.SFSU.txt


====== DICTIONARY 340 C++ =====


------ Keywords: 19
------ Definitions: 61


Search [1]: !help
        |
         PARAMETER HOW-TO,  please enter:
         1. A search key -then 2. An optional part of speech -then
         3. An optional 'distinct' -then 4. An optional 'reverse'
        |
```

```
         Adverb [noun] : Adverb is a word that adds more information about place, time, manner
        |
Search [5]: noun noun reverse
        |
         Noun [noun] : Noun is a word that refers to a person, (such as Ann or doctor), a plac
        |
Search [6]: cSc220
        |
         CSC220 [adjective] : Ready to create complex data structures.
         CSC220 [noun] : Data Structures.
         CSC220 [verb] : To create data structures.
        |
Search [7]: CSc340 noun distinct REVERSE
        |
         CSC340 [noun] : Programming Methodology.
         CSC340 [noun] : Many hours outside of class.
         CSC340 [noun] : A CS upper division course.
        |
Search [8]: |
```

Run: part_c ✕

Reverse [noun] : A dictionary program's parameter.
        Reverse [adjective] : Opposite to usual or previous arrangement.
        Reverse [adjective] : On back side.
        |
Search [15]: reverse distinct reVERSE
        |
        Reverse [verb] : Turn something inside out.
        Reverse [verb] : To be updated...
        Reverse [verb] : Revoke ruling.
        Reverse [verb] : Go back.
        Reverse [verb] : Change something to opposite.
        Reverse [noun] : To be updated...
        Reverse [noun] : The opposite.
        Reverse [noun] : Change to opposite direction.
        Reverse [noun] : A dictionary program's parameter.
        Reverse [adjective] : Opposite to usual or previous arrangement.
        Reverse [adjective] : On back side.
        |
Search [16]: |

Run: part_c ✕

Reverse [verb] : Turn something inside out.
        |
Search [17]: reverse noun ok
        |
        <The entered 3rd parameter 'ok' is NOT 'distinct'.>
        <The entered 3rd parameter 'ok' is NOT 'reverse'.>
        <The entered 3rd parameter 'ok' was disregarded.>
        <The 3rd parameter should be 'distinct' or 'reverse'.>
        |
        |
        Reverse [noun] : A dictionary program's parameter.
        Reverse [noun] : Change to opposite direction.
        Reverse [noun] : The opposite.
        Reverse [noun] : To be updated...
        Reverse [noun] : To be updated...
        Reverse [noun] : To be updated...
        Reverse [noun] : To be updated...
        |
Search [18]: |

```
Run:    part_c ×

        Reverse [noun] : Change to opposite direction.
        Reverse [noun] : The opposite.
        Reverse [noun] : To be updated...
        |
Search [20]: reverse ok ok ok
        |
        <The entered 2nd parameter 'ok' is NOT a part of speech.>
        <The entered 2nd parameter 'ok' is NOT 'distinct'.>
        <The entered 2nd parameter 'ok' is NOT 'reverse'.>
        <The entered 2nd parameter 'ok' was disregarded.>
        <The 2nd parameter should be a part of speech or 'distinct' or 'reverse'.>
        |
        |
        <The entered 3rd parameter 'ok' is NOT 'distinct'.>
        <The entered 3rd parameter 'ok' is NOT 'reverse'.>
        <The entered 3rd parameter 'ok' was disregarded.>
        <The 3rd parameter should be 'distinct' or 'reverse'.>
        |
        |
        <The entered 4th parameter 'ok' is NOT 'reverse'.>
        <The entered 4th parameter 'ok' was disregarded.>
        <The 4th parameter should be 'reverse'.>
        |
        |
        Reverse [adjective] : On back side.
        Reverse [adjective] : Opposite to usual or previous arrangement.
        Reverse [noun] : A dictionary program's parameter.
        Reverse [noun] : Change to opposite direction.
        Reverse [noun] : The opposite.
        Reverse [noun] : To be updated...
        Reverse [noun] : To be updated...
        Reverse [noun] : To be updated...
        Reverse [noun] : To be updated...
        Reverse [verb] : Change something to opposite.
        Reverse [verb] : Go back.
        Reverse [verb] : Revoke ruling.
        Reverse [verb] : To be updated...
        Reverse [verb] : To be updated...
        Reverse [verb] : Turn something inside out.
        |
Search [21]: |
```

■ Does your program work properly?

• Yes.

- How will you improve your program?

    - This was my first decent-sized project with C++, so I have yet to learn about the features of the language. For the most part, I have adapted my solutions from the previous assignment wherever appropriate.

      This version of the program also addresses some edge cases that are outside the scope of the sample output. However, the code quality may have decreased due to unfamiliarity with the language.