CSC 340.01+.02+.03 ASSIGNMENT 03 SUMMER 2025 TA

ASSIGNMENT INSTRUCTIONS

1. Assignment 03: **100 points w/ 0 E.C. points**
2. Due Date & Time: **07-06-2025 at 11:55 PM**

PERFORMANCE TRACKER		
ASMT	**GRADE**	**YOUR GRADE**
CANVAS	05	
01	15	
02	100	
03	100	
TOTAL	220	

A: 90-100% **B**: 80-89% **C**: 70-79% **D**: 60-69% **F**: 0-60%
The course grader provides feedback to your assignments on Canvas.

WHAT TO SUBMIT

1. Assignment Report
2. Code

HOW TO SUBMIT AND THE RULES TO FOLLOW

- The Guidelines for All Assignments
- The Course Policy on Student Conduct and Academic Honesty
- The assignment instructions for this assignment
- The additional instructions provided in class and on Canvas.
- Submit via Canvas, the Assignment Submission section.

ABOUT

- Method vs. Methodology: Method is the tool. Methodology is the justification/ rationale for using a particular method.
- A learning outcome of CSC 340 Programming Methodology is we can recognize a problem, diagnose a problem, define a problem, and formulate a problem. While solving a problem, we frequently take a step back and evaluate available methods.
- Assignment 03 is to provide us with another opportunity to practice these skills.
- All parts of this assignment are to be done in **C++**. Package 04 serves as a reference for this assignment. Please use the package.

Download: http://csc340.ducta.net/Assignments/Assignment-03/Assignment-03-Code.zip
We must use the above-provided starter code to get credit.

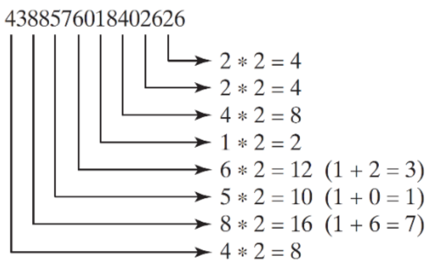
PART A – TIC TAC TOE, **5 points**

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**.
2. Our program must produce **identical** output:
   Assignment-03_PA_Run1.txt and **Assignment-03_PA_Run2.txt**

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.

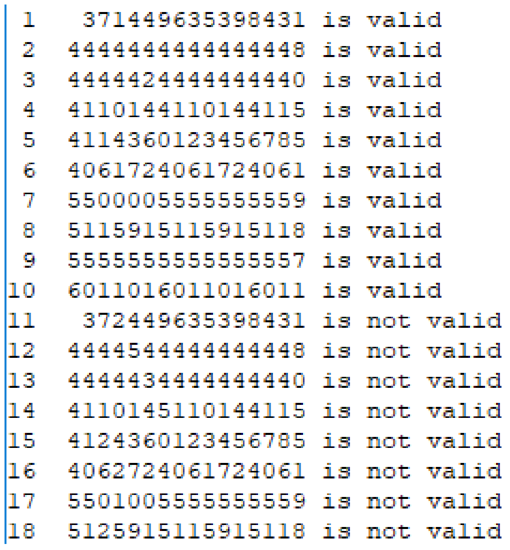


Example: Validating **4 3 8 8 5 7 6 0 1 8 4 0 2 6 2 6**

a) Double every second digit from right to left. If doubling 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.
2. Please do not change the function **main**
3. Your program must produce **identical** output: **Assignment-03_PB_Run.pdf**



```
 371449635398431 is valid
4444444444444448 is valid
4444424444444440 is valid
4110144110144115 is valid
4114360123456785 is valid
4061724061724061 is valid
5500005555555559 is valid
5115915115915118 is valid
5555555555555557 is valid
6011016011016011 is valid
 372449635398431 is not valid
4444544444444448 is not valid
4444434444444440 is not valid
4110145110144115 is not valid
4124360123456785 is not valid
4062724061724061 is not valid
5501005555555559 is not valid
5125915115915118 is not valid
```

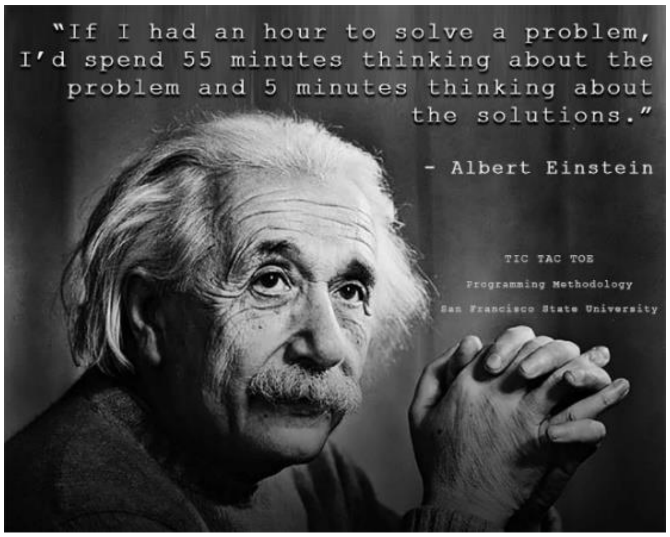
PART C – Dictionary 340 C++, **90 points**

Our satisfied clients are back to ask us to implement another interactive dictionary. Our dictionary takes input from users and uses the input as a 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 the 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 focus on the differences/improvements you are making in this C++ version of the program in comparison to your previous Java version while explaining the following **in detail**:


"If I had an hour to solve a problem, I'd spend 55 minutes thinking about the problem and 5 minutes thinking about the solutions."
- Albert Einstein
TIC TAC TOE
Programming Methodology
San Francisco State University

   - Your analysis of the provided information and the provided complete sample output. *Please think about Clients and Sales.*
   - What problem you are solving. Please explain it clearly then define it concisely. *Please think about Problem Solving and Interviews.*
   - How you store data from the external text file. And why. *Please think about Data Structures and Data Design.*
   - Which data structures you use/create for your dictionary. And why. *Please think about Data Structures and Data Design.*

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

   - Implement your program to meet all the requirements.
   - In your assignment report, demonstrate your program to your grader/client.
   - Does your program work properly?
   - How will you improve your program?

Happy problem-solving and happy coding!