CSCI-1200 Data Structures — Fall 2025
Homework 0 — Guessing Game

Before starting this homework, please read the Syllabus and Collaboration Policy and complete the short online quiz before starting the first homework.

This homework is a warmup for you to practice C++ programming with command line arguments, printing, reading input from a text file, using random numbers, and the C++ STL `string` and `vector` classes.

Please read the entire handout for the assignment before starting to program. In addition to the lecture notes, you will also want to refer to the

- "Helpful C++ Programming Information" and
- "Good Programming Practices"

sections of the course webpage.

You will create a simple text-based, interactive number guessing game. An example is shown on the right. The program will select at random a secret number from an input file specified on the command line. The program will tell the user how many digits the secret contains. Then the user will attempt to guess the number, entering a string of the correct length and pressing return. After each guess the program will use *ascii art* to visualize if any of the digits in the guess are correct.

If the digit is in the secret but not in the correct location, it will be surrounded by the period '.' character. If the digit is in the correct location, it will be surrounded by the asterisk '*' character. When the user guesses the complete and correct secret, the program prints a simple message with the number of guesses used and the program exits.

The secret message may contain duplicate characters. Note the behavior when the guess contains multiple copies of a digit, but the secret only contains 1 copy of that digit. If the user guesses the correct location of the digit, that position is highlighted with asterisks and other copies of the digit are not highlighted. If none of the positions of that digit are correct, only the first instance of the digit is highlighted with periods.

```
Let's play a guessing game!
The secret is a number with 5 digits.
your guess: 01234

+-----------------------------+
|..... ..... .....           |
|.   . .   . .   .           |
|. 0 . . 1 . . 2 .   3     4 |
|.   . .   . .   .           |
|..... ..... .....           |
+-----------------------------+

your guess: 56789

+-----------------------------+
|                 *****       |
|                 *   *       |
|  5     6     7  * 8 *   9   |
|                 *   *       |
|                 *****       |
+-----------------------------+

your guess: 22001

+-----------------------------+
|      ***** .....       .....|
|      *   * .   .       .   .|
|  2   * 2 * . 0 .   0   . 1 .|
|      *   * .   .       .   .|
|      ***** .....       .....|
+-----------------------------+

your guess: 12180

+-----------------------------+
|***** ***** ***** ***** *****|
|*   * *   * *   * *   * *   *|
|* 1 * * 2 * * 1 * * 8 * * 0 *|
|*   * *   * *   * *   * *   *|
|***** ***** ***** ***** *****|
+-----------------------------+

CONGRATULATIONS!
You solved the puzzle after making 4 guesses!
```

Command Line Arguments

Your program will expect a single argument on the command line, the name of a file containing one or more candidates for the secret. Here are a few examples of how you can run your program:

```
./game.out powers_of_two.txt
./game.out 1000_5_digit_numbers.txt
./game.out troy_zip_code.txt
```

The provided sample input text file '`powers_of_two.txt`' contains the first 24 powers of two, each on its own line. '`1000_5_digit_numbers.txt`' contains a thousand 5 digit numbers, each on its own line. Your program will *randomly* select one of the numbers from the file as the secret. It can be helpful when you are getting started with development and also when you are testing and debugging the program to have a consistent secret, so we suggest using a file like '`troy_zip_code.txt`', which contains a single number.

Expected Output and Viewing ASCII Text Files

You must exactly follow the specifications for the command line and output to ensure you receive full credit for your work. The provided sample output files and the validation script on Submitty will also help you check your work.

Make sure you're using a good file viewer/editor to look at the provided input and output files. It should correctly display the UNIX/GNU Linux '`\n`' line ending; good code viewers/editors are listed on the "Editors, Compilers, and IDEs" page. Don't attempt use the Windows line ending '`\m`' or '`\r`' characters on any homework for this course, because this will fail validation tests on Submitty.

Additional Requirements

This is a simple program and you should write all of your code in a single `.cpp` file. However, don't write all of your code inside the `main` function. You should organize the logic into multiple functions. You should use the STL `string` and `vector` classes that have been / will be covered in Lectures 1 and 2 and Lab 1. Please don't write new classes or use concepts that we haven't yet covered in lecture/lab.

Use good coding style when you design and implement your program. Review the "Good Programming Practices" section on the course webpage to be sure that the TAs will be able give you credit for your hard work. Use good variable and function names, and don't forget to comment your code! Be sure to test your program with a variety of different input files, including files that you prepare yourself!

Download and fill out the provided template `README.txt` file, adding any notes you want the grader to read. You must do this assignment on your own, as described in the "Collaboration Policy & Academic Integrity" document. Be sure to list the names of anyone you talked to about the problem or debugging and all references you consulted in preparing your solution.

More About Homeworks and Submitty

Homework instructions are generally posted on Friday and you will have a week to work on the assignment before it is due. The Submitty configuration for submission and autograding is generally made available sometime on Monday.

You are encouraged to submit your assignment early – to verify that you understand the assignment instructions and are making good progress. Carefully examine the autograding results, correct any mistakes, and resubmit as needed. You may submit up to 20 times with no penalty.

Getting Help

If you have questions about the assignment or would like help on your development and debugging process, please attend "TA and Mentor Office Hours". You may also ask short questions about the homework and lecture material on the "Discussion Forum".