|  |
| --- |
|  |
| Project 1 |
| Mastermind |
|  |
| **Joseph Levin CSC-17A**  **Section 43950**  **Spring 2015** |
| **5/4/2015** |

|  |
| --- |
| This write up contains an introduction which consists of a description of Mastermind and why this game was chosen for the project. It summarizes the project with statistics including satisfaction of criteria, what variables, constructs, and processes were implemented, and notes on exceptional areas of the project(i.e, challenges encountered during the development process). It describes the development process with example inputs/outputs, flowcharts and pseudocode, and details of major variables. The write up lists which concepts were used from the textbook, and finally includes the documented code. |

|  |  |
| --- | --- |
|  | Table of Contents |
| 1 | Introduction |
| 2 | Summary   1. Program Statistics 2. Areas of Note in Development |
| 3 | Description   1. Program Walkthrough 2. Flowcharts 3. Variables/Structures 4. Concepts Implemented |
| 4 | References |
| 5 | Code Documentation |

1. Introduction: Mastermind

Mastermind is a two-player board game in which turns are taken by alternating between the roles of a *codebreaker* and a *codemaker*. The codemaker chooses a pattern—be it a pattern of colored marbles or simply numbers—and the codebreaker attempts to guess the pattern within a limited number of turns. The codebreaker is given clues in the form of being told how many correct elements (i.e, red marbles or the number three) his guess has as well as how many of those elements are in the correct position of the pattern. It’s important to note that a guess could include all the right colors or numbers, but none of them could be in the right position.

The game of Mastermind makes a good fit for a computer game because the role of the codemaker is easily adapted to be played by a computer. A computer is perfectly capable of coming up with random patterns, and is just as fit at telling a person how close to that pattern they are. A computer would also never take out its frustrations on the codebreaker if he/she demonstrations proficiency at decipher its patterns. As a matter of fact, a computer may be preferable to playing with a friend, if one has lousy friends.

Mastermind is an outwardly simple game. Given a secret pattern, figure out the pattern…it is the kind of thing a person might accidentally get correct without realizing they were playing a game in the first place. However as with any good game, it offers an experience that leaves the player satisfied and feeling good. Everybody likes being correct, especially when it comes packaged with a slight taste of having bested someone. With Mastermind, the codebreaker is given just this kind of opportunity.

1. Summary

|  |  |
| --- | --- |
| 2.1 Statistics | |
| Program Length (lines of code). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 479 |
| # Structures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 4 |
| # Variables . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 32 |
| # Functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 16 |

* 1. Areas of Note in Development

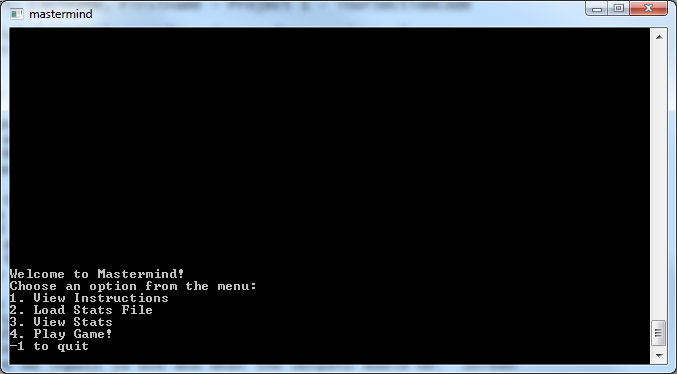
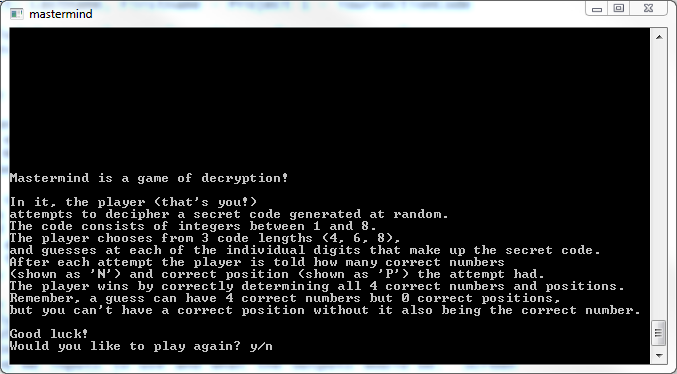
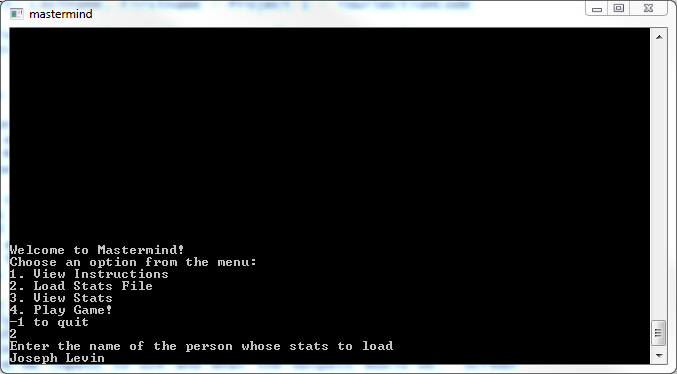
At the end of the day the most significant obstacle in the development of the project was my own indecisiveness. I debated with myself over a few different ways of implementing how to handle a user’s guesses as well as where to keep track of certain elements such as how many attempts the user has taken. I settled on the current iteration because I felt the structure-within-structure approach suited the logic behind a user’s guesses very well. It also allowed for a simple system for keeping track of accuracy of guesses, and comparing how many guesses have been taken to how many are allowed. Also, structures were very beneficial as it made implementing a form of stat tracking almost trivial when combined with binary file I/O.

The logic I used for checking a guess against the correct pattern feels sort of gimmicky. I would have preferred to not rely as heavily on copying to temporary arrays in order to do my checking, and my solution of reassigning the elements in the temporary arrays to a negative to indicate it has already been checked feels very brute-force. However a more elegant process was and is lost to me.

From initial planning to typing this write up, the project took about a week and a half to complete. There are some features that I experimented with such as save files that I was unable to implement into the final project. Given the chance, I would like to revisit them.

1. Program Description

3.1 Program Walkthrough



The main menu offers 4 options to the user.

1. View Instructions
2. Load Stats File
3. View Stats
4. Play Game

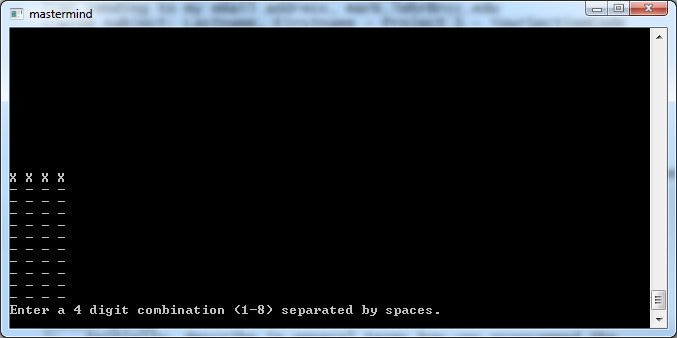
And also it gives -1 as the option to quit

If the user inputs 1, they are shown a brief explanation of how Mastermind is played. It should be noted that the original game allows 6 different colors, but I forgot this and chose 8 instead.

Choosing 3 prompts the user to enter the name of a stats file to load.

Error displayed if not found

If found, file is read into a Stats struct via binary file IO



Choosing 3 displays the contents of a loaded Stats struct

Default values are all zero

Displays error if all zeros due to division by zero

Choosing 4 launches the game

Player is prompted to choose code length

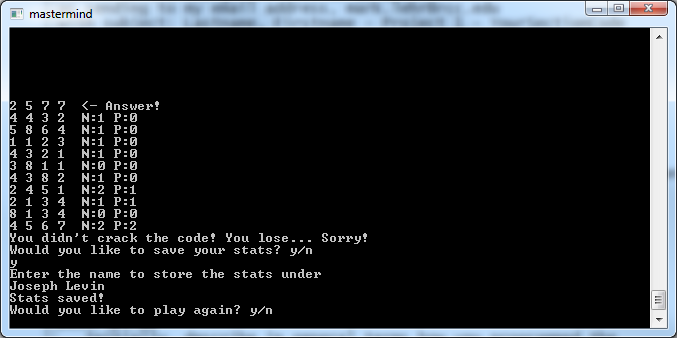
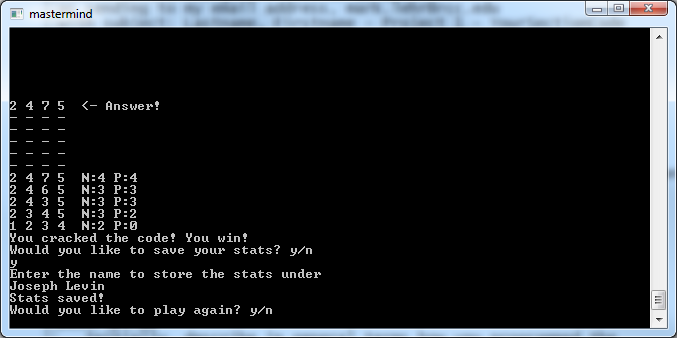
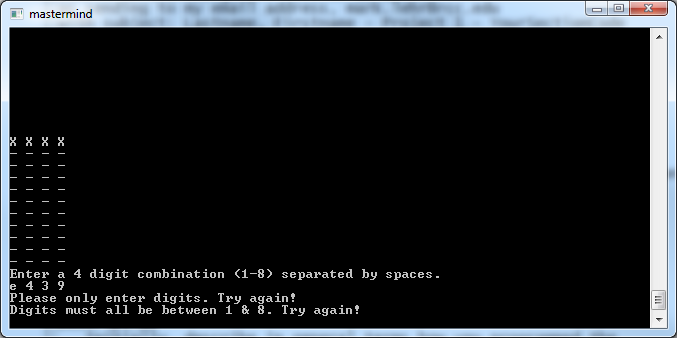
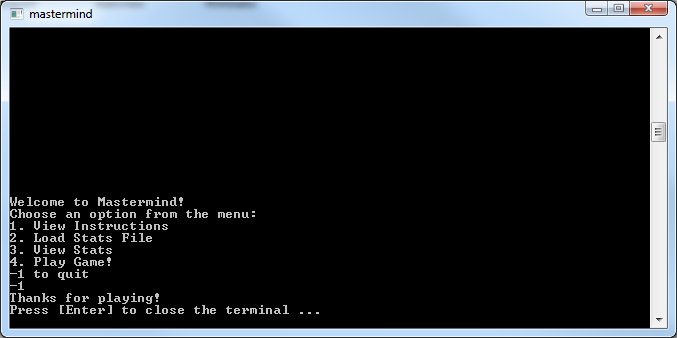
Note: if length=l, then odds of getting right answer first try is 1/(8^l).

i.e, l=4, odds=1/(8^4)=0.0002

After choosing length (4) board is displayed

Input will only accept 4 digits separated by spaced

i.e, input = 1 2 3 4



Upon correctly guessing, answer is revealed and a victory message is displayed

User is prompted to save stats

i.e, input y to save

input <file name>

Similarly, losing reveals answer and a defeat message is displayed

User is still prompted to save stats

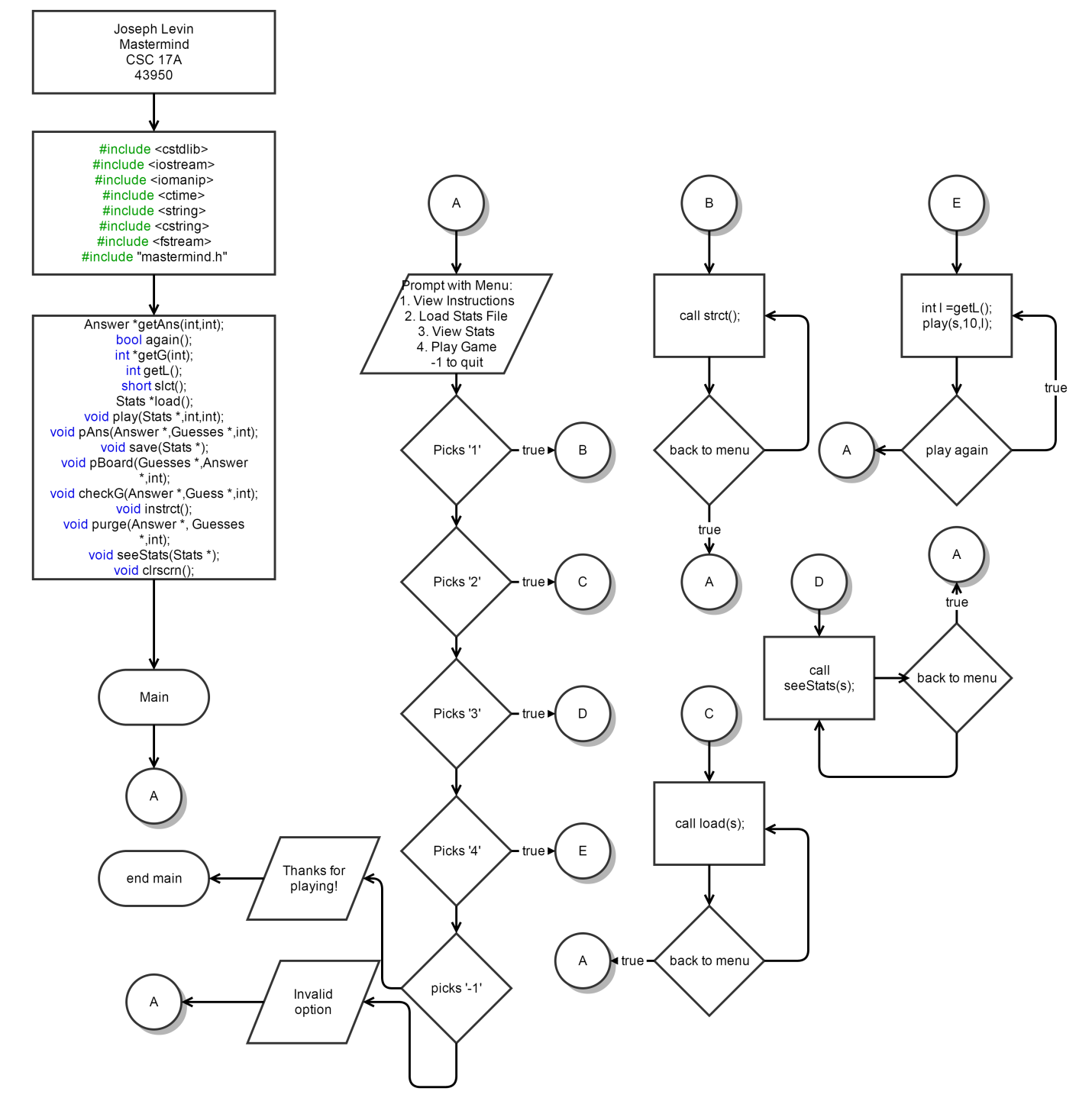
If file name is reused, old stats are overwritten

Input will only accept digits 1 – 8.

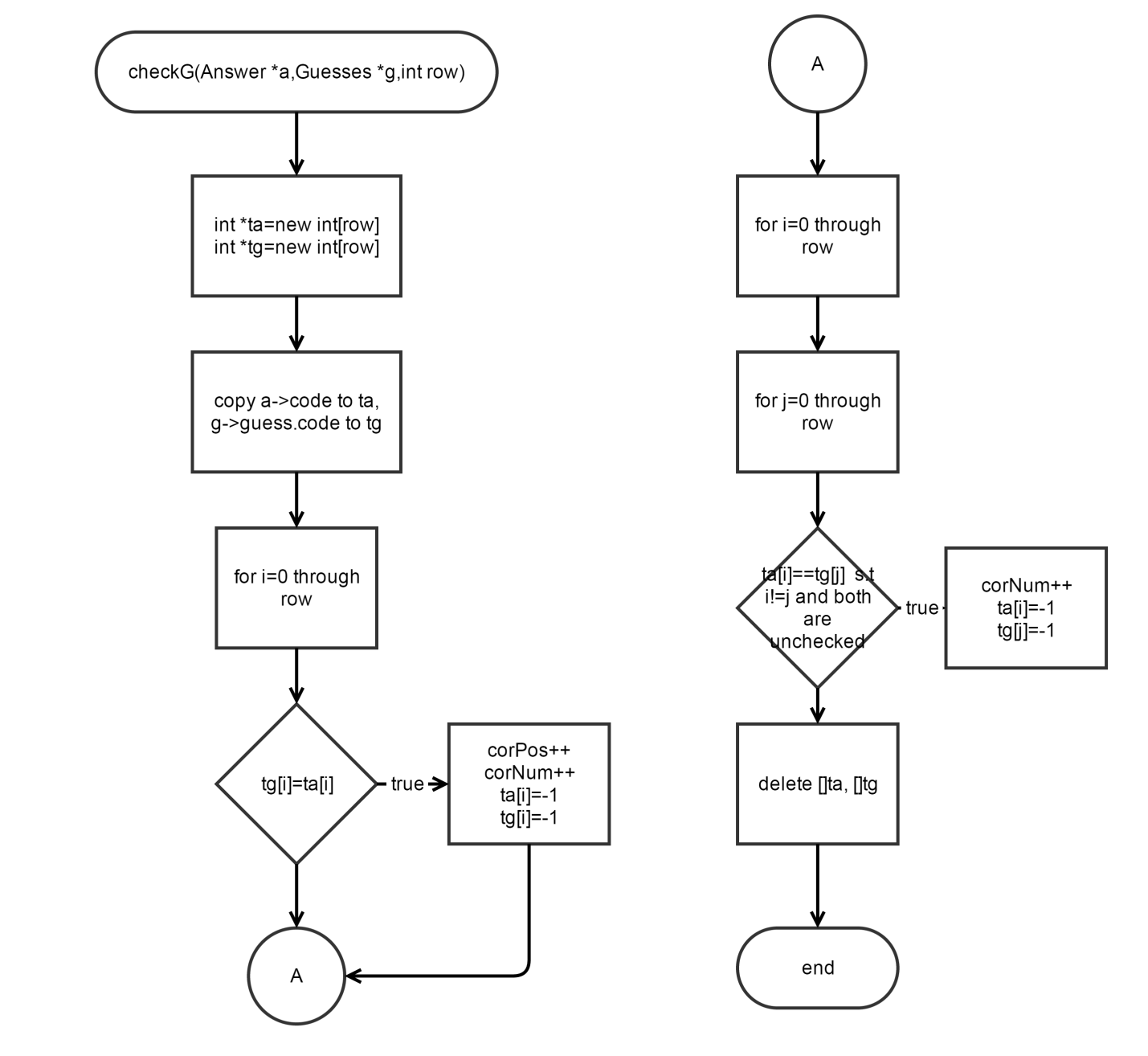
Non digits will give appropriate error, as will digits outside of range

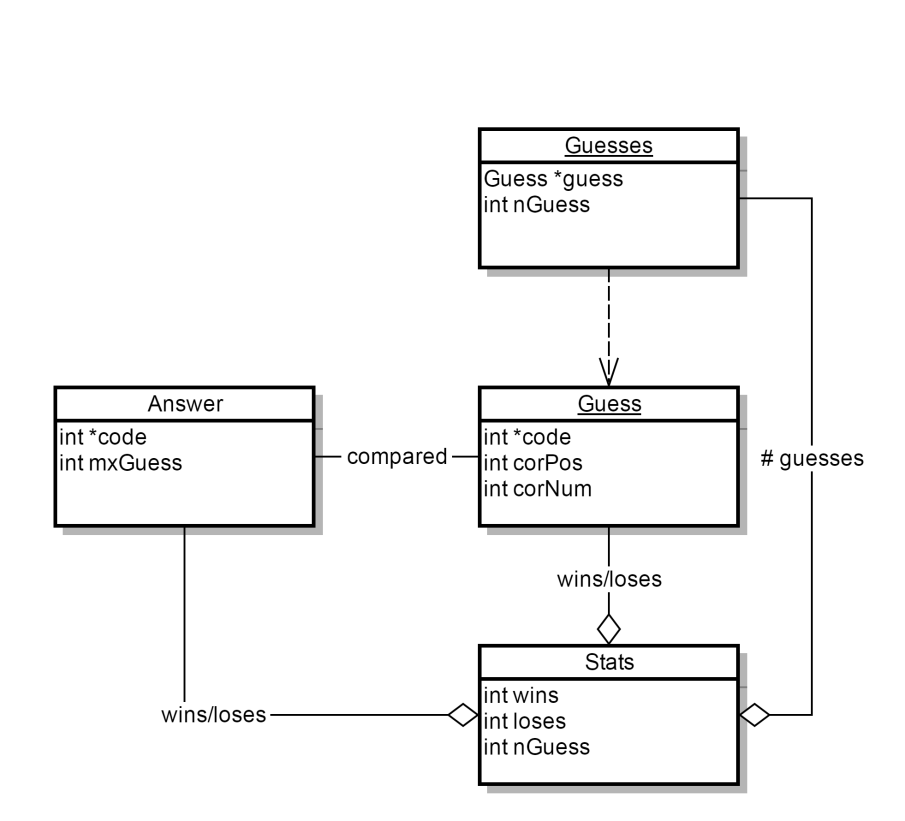
Choosing -1 in menu quits program

* 1. Flowcharts

Main menu:

Guess Checking Function (checkG):



* 1. Variables/Structures
     1. Structure UML:

* + 1. Structures/Variable Listing:

|  |  |
| --- | --- |
| Answer | int \*code  stores answer pattern  int mxGuess  stores maximum guesses allowed |
| Guesses | Guess \*guess  stores individual guess structures (for each turn)  int nGuess  stores number of guesses taken |
| Guess | int \*code  stores guess attempt  int corPos  # of positions guessed correctly during a turn (determines win)  int corNum  # of numbers guessed correctly during a turn |
| Stats | int wins  stores number of games won  int loses  stores number of games lost  int nGuess  stores total guesses taken between all games |

* 1. Concepts Implemented: