CS205 — Software Engineering

Warm-up Project: Simulation of the board game Mastermind

Individual Report

1. Summary

This project implements a Mastermind simulation program with three main functions. First, user guesses the code randomly generated by computer within 12 turns. Second, computer guesses the code generated from user's input within 12 turns as well. Third, the simulation program provides statistics of user's and computer's winning history, such as the average and the best record of turns and duration. Statistics also includes record of each winner's name which is typed in after a user finds the code.

The major structure of this program was discussed by both team members. Then, one was responsible for the interface and user mode except the checking function, and the other worked on that function. After user mode was finished and integrated with the main menu, one member implemented the statistics, and the other completed the computer mode based on the algorithm provided on Wikipedia.

2. Development

a. Description of what the system does

The system provides a command-line based mastermind game. The rule of it refers to the Mastermind page on Wikipedia. A guess and an answer both contain four pegs of six possible colors in order. After a guess is entered to the system, the system returns a key containing at most four black or white pegs. A black peg refers to a peg

which is correct in both color and position. And a white peg indicates that a peg is correct in color but placed in wrong position. Players need to make a correct guess and receive four black key pegs within twelve turns. The system also provides features that user guesses the answer generated by computer and that computer guesses the answer typed in by user. In addition, the statistics collects and summarize the number of used turns and duration of each winning history.

b. How the simulation engine works

In the main menu, user selects one of functions, including user mode, computer mode, statistics, and quit. First, in user mode, user guesses the code randomly generated by computer. User repeats typing in four pegs in possible colors and receiving a key with black and white pegs within twelve turns. User can also request hints for the color of a peg. Because user can type in colors in four separate lines, there is a feature of clearing current guess and restart this turn. Second, in computer mode, computer guesses the code generated by user. In the beginning, user decides the answer key by inputting four pegs in possible colors just like the way in the user mode. Then, computer guesses the answer based on the five-guess algorithm. Third, in statistics mode, the system demonstrates winning record of users and computer, including average and minimum time and number of turns used to win.

c. What the components of the application are and what part they played in the development of the system

First, the functioning application covers the whole program, which is written in three python files: *mastermind*, *codecheck*, and *computerguess*. The *codecheck* includes a function to return key pegs according to the input guess and answer. The

computerguess takes the answer and starts the guess by computer. And the *mastermind* covers the rest of functions. Second, the user interface is completed in simple input and print functions all over in the program. Third, the facility to gather statistics of winning history is provided as a mode like user mode or computer mode.

d. What part the other members of the team played in the development

Junxiao implemented the function of receiving guess and answer and returning key. Also, he developed the computer mode with the five-guess algorithm.

e. How the software was tested

The software was first tested after we integrated the interface, input and output in user mode, and the check function. In this phase of testing, we just checked if our programs could be combined properly. Then, it was second tested after we finished the computer mode and statistics. In this phase, we played this game for several times, examined the correctness of the application, and gathered enough data for statistics. Afterwards, it was tested again after we finished all implementation and started preparing the presentation to see the flow of the game. In this phase, we tested the game like a new user and played around all functions.

f. How much time and effort was spent on the four activities of software development (specification, development, testing, and deployment)

We spent about two hours on specification in two meetings. During the first meeting, we listed the requirements of the game and designed the structure. This might take a little longer than the second meeting. We talked about our opinions of each other's work and planned some change we needed to make during the second meeting. We spent five to six hours individually on our own part of program approximately.

Testing may have taken one and a half hours. We run the program for several times and tried to find bugs to fix. We spent around one hour on deployment, which means showing this game to our friends and letting them play the game.

3. Analysis

a. Describe which activities were accomplished easily, and which were difficult and why

Gathering the requirements was the most easy activity in this project, because there are clear instructions of mastermind and many existing online games familiarizing us with the rule. Designing the structure of this application was a little hard, because we didn't have the detailed requirement of interface and procedure. Division of labor was also difficult, because we had to list clear and separated enough tasks to assign work to each other. In addition, integration of different tasks was challenging, because we used different format of output and different naming styles.

b. How members communicated with each other and whether this communication was effective

We discussed simple issues and reported how our work was done via Wechat. This was effective because we could communicate whenever we needed and whenever we were free. Github was used to manage our programs and control versions. This was also effective because we could store our code online and share it easily. We also had meetings either in our apartment or around campus.

c. What errors were uncovered during testing

We forgot to provide hints in user mode, so we implemented it after we finished most work. To make the winning history cover the number of hints used in a game, this attribute was added to the statistics very late. Therefore, there was a key error when we run the statistics because the file storing history data was old and not covered that attribute.

d. Whether development actually proceeded as planned

We made our schedule and tried to follow the plan, but we delayed our meeting a little, because something came up and we did not have enough time on the program.