COS 125 Testing Document

Testing Document
FIVE DICE, THREE ROLLS
Spencer Ward
October 14, 2016

SECTION 1. INTRODUCTION

Five Dice, Three Rolls is a command line version of the classic dice game, Yahtzee. The game will always display the dice and the scoresheet, and the users will interact with the game through text prompts. The game will support any number of players, but is best played in a small group. The target audience will be elementary school students and up, as long as they know how to type in the command line and can understand the rules of Yahtzee. The game will be meant for entertainment, mostly in small groups.

SECTION 2. OVERVIEW OF THE GAME

The game is played in a series of thirteen rounds, with the players choosing one of the thirteen scoring types for each round, without repetition. Every round, each player rolls five dice, then selects as many dice as they want to reroll, and then rerolls again. Their score in the scoring type they chose is recorded in a score table. At the end of the thirteen rounds, the sub scores are summed up, and the player with the highest total wins the game. Scores and dice are displayed in text. The game does not support saving, high scores, or sound.

SECTION 3. TOP LEVEL FAILURE MODES

The most dangerous point of failure for the game is going to be the `raw_input()` based user input. During the setup process, the players are asked a series of questions that are going to expect integer responses (e.g. "How many people are going to play?"), so the code behind this section is going to have to check that the responses are integers that are within reasonable boundaries. If they are not, it should print a short message saying that the input doesn't make sense, then prompt the input to give a response again. Later in the program, the user is going to be asked to select dice to reroll, rules to apply to their dice, and for a prompt on switching players. Each of these is going to need to check that the validity of the response, then give a brief message and ask again if the input does not make sense. When selecting dice, the program is going to need to ask for a specific format for the list of dice to reroll, then check that the formatting is followed before it can try to interpret the input. Once these basic conditions are tested, the program can be run with various types of inputs, including phrases, special characters, blank strings, and unexpected formats.

SECTION 4. MODULES AND TESTING

The only modules that will be used in the game are random.py and os.py. Both of these modules are going to be hardcoded with working code, so there should be no way for these modules to run into errors under normal conditions. A potential problem that might come up, however, is if the code is run on different platforms. The method the game will use to clear the screen is to use the `system(<command>)` method in os.py to run the system command to clear the console. As different platforms have different command sets, the game will need to check the system operating system to determine which command to use for clearing the screen. The program can run `cls`, and `clear` in Linux.

SECTION 5. TOP LEVEL TESTING

Testing for the game can consist of trying to test each of the input stages with unusual inputs, including long phrases split up with multiple spaces, numbers where the game expects strings, and strings where the game expects numbers. Other tests include intentionally scoring 0 points to test for boundary conditions, trying negative values while picking dice, and experimenting with unusual (negative, large, or 0) inputs during the setup stage.

Once this is finished, I will try playing it with a friend to see if they can find ways to break it. I will take note of any bugs that they find and fix them before the next game, if possible.

SECTION 6. IMPLEMENTATION PLAN AND TIMELINE

For each version of the game I complete, I will set aside some time for testing.

The first version of the game will mostly be based around the UI, so I will spend up to half an hour trying different parts of the UI and making sure that everything is printed properly, ASCII art scales as it should, and everything looks fine in different monospace fonts.

The second version will be focused around user input, so I will spend at least half an hour testing different inputs and checking that the user inputs and UI redrawing function properly.

The third version will implement dice rolling, rounds, and scoring. Dice rolling should take around fifteen minutes to test, and I will spend another fifteen minutes making sure that everything progresses as it should as the game plays out.

The fourth version will be based around implementing the thirteen different scoring types, which I will aim to spend half an hour to an hour testing different conditions for. I will try setting different dice combinations to make sure that they always output the correct scores, and test different inputs to make sure that unusual inputs will be handled without crashing the game.

Once the game is completed, I will spend half an hour on my own testing different conditions and making sure that nothing from previous versions was broken when implementing more recent ones. If everything goes smoothly during this section, I will spend an hour letting other people try it out, and use their feedback to fix any problems that might be left in the code.

SECTION 7. BIBLIOGRAPHY

No outside sources used.

SECTION 8. WORK ASSISTANCE STATEMENT

No outside assistance was used.