GuessMasterPro - Final Project Report

GuessMasterPro is a word guessing game you play in the terminal. The goal is to figure out a hidden word by guessing one letter at a time. I got the idea to make this game from everything we've learned in class. The game shows you which letters you've already guessed, how many chances you have left, and keeps track of your score. You earn points for correct guesses and lose points for wrong ones. It's a fun and simple way to use what we've learned—like how to work with loops, user input, and different data structures. It also makes you think and test your memory, logic, and vocabulary while playing.

Algorithm #1: Letter Guessing

Steps:

1. Convert guessed letter to lowercase.
2. Check if it's a letter and hasn't already been guessed.
3. Add it to the set of guessed letters and push to history stack.
4. If the letter is in the word, increase score.
5. If not, decrease score and guesses.

Code Snapshot:

```
makeGuess(char letter)

if (hiddenWord.contains(String.valueOf(letter))) {
   score += 10;
} else {
   remainingGuesses--;
   score -= 5;
}
```

Algorithm #2: Win Check

Steps:

1. Loop through each letter in the word.
2. Check if it exists in the guessed letters set.
3. If any letter is missing, return false.
4. If all are present, return true.

Code Snapshot:

```
hasWon()

for (char c : hiddenWord.toCharArray()) {
   if (!guessedLetters.contains(c)) {
      return false;
   }
}
return true;
```

Algorithm #3: Score Tracker

Steps:

1. Initialize score at 0.
2. On correct guess, add 10 points.
3. On incorrect guess, subtract 5 points.
4. Return score using getScore().

Code Snapshot:

```
getScore()

return score;
```

3. Explanation of Algorithms

The Letter Guessing algorithm is the heart of the game. It takes whatever letter the player types in, checks if it's a valid guess, adds it to the list of guessed letters, and updates the score depending on if it was right or wrong. I built this from scratch using basic Java tutorials and by thinking through what should happen when someone makes a guess—like if they repeat a letter, guess wrong, or get it right. The Win Check algorithm was made to figure out if the player has guessed the whole word. It goes through each letter in the hidden word and sees if the player already guessed it. If they missed even one, the game keeps going. I made sure this part runs quickly so the game works well even with longer words. The Score Tracker was a fun extra feature I added after brainstorming ways to make the game more exciting. It gives points for correct guesses and takes away points for wrong ones. I used a simple number counter that goes up or down depending on how the player does. It adds a bit of pressure and fun to the game.

4. Time Complexity of Algorithms

- Letter Guessing: O(n), where n is the length of the word, because it may check for the letter in the word.
- Win Check: O(n), loops through each letter of the word to see if it's guessed.
- Score Tracker: O(1), since it's just math operations on an integer.

5. Data Structures Used

- HashSet<Character>: Used to store guessed letters. It was chosen because lookup time is fast (O(1)), which is ideal for checking repeated guesses.
- Stack<Character>: Used to store guess history, which sets the foundation for an undo

feature in future versions.
- String: Stores the hidden word. Simple and efficient for character comparison and display.

6. Error Fixing & Future Improvements

While working on the game, I realized it would be more exciting if players had a way to track their performance. That led to adding the score system. It was a simple but powerful change that made the game feel more rewarding and goal oriented. One problem I ran into was that the game allowed users to guess the same letter multiple times, which was frustrating. To fix this, I added a check in the makeGuess() method to return false if the letter had already been guessed. This improved the game's fairness and functionality. In the future, I would like to add a full menu system, the ability to play multiple rounds, a hint system that reveals one letter, and an undo feature using the stack. This would make the game feel more complete and polished, while using the existing data structures in new and creative ways.