1. **(10 points) Chapter 5: Self-check problems #1, #3, #12, #14 and #20.**

   For your benefit, I suggest you complete all of the self-check problems on your own. However, I will just collect the answers to the above five problems. Please type the answers into a document.

   a. NO NEED to copy original questions, just put down the question number and the answer.
   b. Name the file as "***LastnameFirstname5.doc***" (or ".docx" or ".pdf", where "Lastname" is your last name, and "Firstname" is your first name), submit it online.

2. **(40 points) Chapter 5: Programming Project #3 "Guessing game"**

   a. Download the attached "`GuessingGame.java`" file.
   b. Modify it so that the program plays a number guessing game with the user.
      i. First, the program picks a random integer between 1 and 100 (inclusive), then it accepts guesses from the user until the user guesses the number correctly. After each incorrect guess, the program will tell the user whether the correct answer is higher or lower than the guess. Once the user types the correct number, the game ends and the program reports how many guesses were made. An example game is shown below:

```
I'm thinking of a number between 1 and 100...
Your guess? 50
It's lower.
Your guess? 25
It's higher.
Your guess? 35
It's lower.
Your guess? 30
It's higher.
Your guess? 32
It's lower.
Your guess? 31
You guessed it in 6 guesses!
```

      ii. After each game ends and the number of guesses is shown, the program asks the user if he/she would like to play again. Assume that the user will type a one-word string as the response to this question. A new game should begin if the user's response starts with a lower- or upper-case Y. For example, answers such as "y", "Y", "yes", "YES", "Yes", or "yeehaw" all indicate that the user wants to play again. Any other response means that the user does not want to play again. For example, responses of "no", "No", "okay", "0", "certainly", and "hello" are all assumed to mean no.

iii. Once the user chooses not to play again, the program prints overall statistics about all games played: the total number of games, total guesses made in all games, average number of guesses per game (as a real number **rounded to the nearest tenth)**, and best game (fewest guesses needed to win any one game) are displayed.

Here is an example of the entire (multiple) game play:

```
I'm thinking of a number between 1 and 100...
Your guess? 50
It's higher.
Your guess? 75
It's lower.
Your guess? 65
It's lower.
Your guess? 64
You guessed it in 4 guesses!
Play again? YES

I'm thinking of a number between 1 and 100...
Your guess? 37
You guessed it in 1 guesses!
Play again? y

I'm thinking of a number between 1 and 100...
Your guess? 50
It's lower.
Your guess? 25
It's lower.
Your guess? 13
You guessed it in 3 guesses!
Play again? no

Your overall results:
Total games = 3
Total guesses = 8
Guesses/game = 2.7
Best game = 1
```

iv. Assume valid user input. When prompted for numbers, the user will type integers only, and they will be in proper ranges. Read user yes/no answers using the Scanner's `next` method (not `nextLine`, which can cause strange bugs when mixed with `nextInt`). To test whether the user's response represents yes or no, use String methods seen in Chapters 3-4 of the book. If you get an `InputMismatchException`, you are trying to read the wrong type of value from a Scanner.

v. For this assignment you are **limited to the language features in Chapters 1-5** of the textbook.

vi. Start simple -- I suggest that you begin by writing a simpler version that plays a single guessing game. Ignore other features such as multiple games and displaying overall statistics. It will be much easier to add those in after you make the single game work as expected.

vii. Remember, your program will be graded both on "external correctness" and "internal design and style" (whether your source code follows the **style guide**).

c. Submit the final "`GuessingGame.java`" file **(DO NOT change the file name)** online.