CSE143X—Computer Programming I & II
Programming Assignment #3
due: Friday, 10/15/21, 11 pm

This assignment will give you practice with if/else constructs, while loops, strings, and pseudorandom numbers. You are going to write a program that allows the user to play the game Bagels, a variant of the board game Mastermind. In this game, the computer generates a random number that the user has to guess. The computer provides hints based on the value and position of digits guessed. The solution will always be a number made up of the digits 1 through 9 (no zeros).

After displaying an introduction, your program should allow the user to play a series of games and should report some overall statistics when the user no longer wants to play again. For each individual game, your program should repeatedly prompt the user for guesses and provide a clue after each incorrect guess. If no digits match between the guess and the answer, the program prints "bagels." The program prints "fermi" for each correct digit in a correct position and "pico" for each correct digit in the wrong position. All of the fermi clues are reported first so as not to give any extra information to the player.

You should study the sample logs of execution at the end of this writeup and on the class web page because your program must exactly match the behavior and format of these logs.

At a minimum, your program should have the following static methods in addition to method main:

- a method that introduces the game
- a method to play one game with the user (just one game, not multiple games)
- a method to report overall results to the user

You should introduce other methods as well so that no single method has more than 25 lines of code. You are encouraged to include the replace and round1 methods discussed in lecture.

When you ask the user whether or not to play again, you should use the “next()” method of the Scanner class to read a one-word answer from the user. You should continue playing if this answer begins with the letter “y” or the letter “Y”. Notice that the user is allowed to type words like “yes”. You are to look just at the first letter of the user’s response and see whether it begins with a “y” or “n” (either capitalized or not) to determine whether to play again.

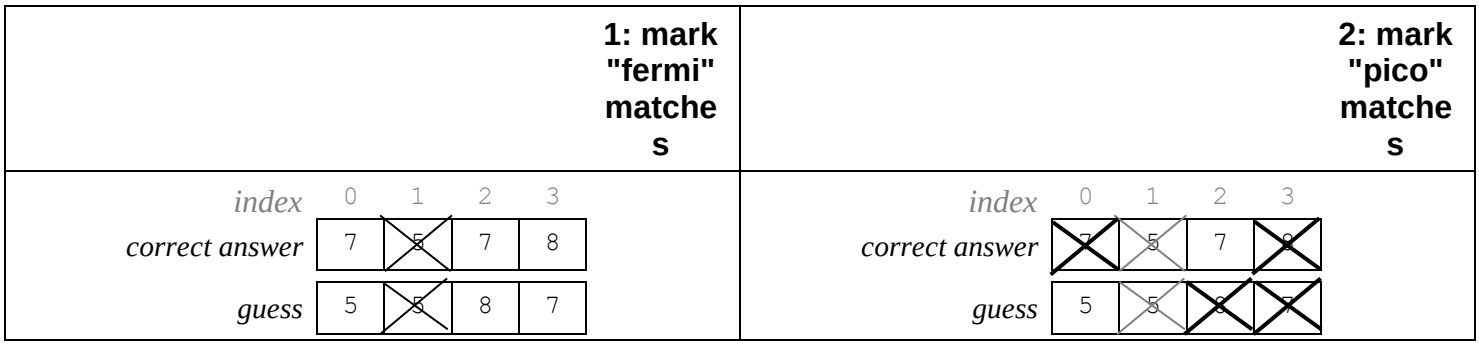
Assume that the user always gives an appropriate value for the length of the answer (an integer between 1 and 9), types a legal value when guessing (all digits, correct length, no zeros), and that the user gives you a one-word answer beginning with “y”, “Y”, “n” or “N” when asked whether to play again. You may assume that no game involves more than 9,999 guesses.

You will notice at the end of the logs that you are to report various statistics about the series of games played by the user. You are to report the total number of games played, the total number of guesses made (all games included), the average number of guesses per game, and the best (fewest) number of guesses used in any single game. The average number of guesses per games should be rounded to one decimal place (you can use either the round1 method or a printf).

Most Java programmers would solve this problem with arrays, but you aren’t allowed to solve it that way. Instead, you will be manipulating strings. You should construct an appropriate string as the answer and you should read the user’s guess as a string using the next() method of the Scanner class. You will find it helpful to use the replace method discussed in lecture that allows you to construct a new string that replaces one character of a string with a different character. You will generate clues by manipulating string variables.

To determine fermi and pico hints, we recommend doing two passes over the digits: the first one to identify fermi matches (matching digits at matching locations) and the second one to identify pico matches (matching

digits at different locations). You must be sure that the same digit is not used for two different matches. You will therefore need to mark digits as "used" so that they will not be used again in another match. For example, in the figure below, the numbers with an 'X' through them should no longer be considered (correct is "7578" and guess is "5587"):



There was one fermi match and two pico matches, so the correct clue would be "fermi pico pico."

Here are a few helpful hints to keep in mind.

- To deal with the yes/no response from the user, you will want to use some of the String class methods described in section 3.3 and 4.1 of the book. You should use the Scanner's next() method to read a word from the console and nextInt() to read simple ints.
- Because this program uses pseudorandom numbers, you won't be able to recreate this exact log. The key requirement is that you reproduce the *format* of this log and that your calculations for overall statistics are correct for your log.
- While you are developing your program, you might want to have it print out the answer before the user begins guessing. Obviously you don't want this in the final version of the program, but it can be helpful for you while you are developing the code.
- The chapter 5 case study is a particularly relevant example for this assignment.

You should handle the case where the user guesses the correct number on the first try. Print the following message:

```
You got it right in 1 guess
```

As noted above, you are to break up your program into methods so that no single method has more than 25 lines of code. You can also include more code in your main method than we normally allow. In particular, you are required to have a while loop in main that plays multiple games and prompts the user for whether or not to play another game.

We will once again expect you to use good programming style and to include useful comments throughout your program. We will expect you to make appropriate choices about when to store values as int versus double, which if/else constructs to use, what parameters to pass, and so on. You are not allowed to use a break statement and you are not allowed to use a return from a void method for this or any future 143X assignment.

To make it easier to check your output, we are requiring that you put together your answer in a particular way. You should construct a new Random object each time you generate a number to be guessed, you should generate one digit at a time, and you should put the answer together in forwards order (1st digit generated is 1st digit of answer, 2nd digit generated is 2nd digit of answer, etc.). You can test your solution by constructing your Random object with the seed 42, but don't include this value in the version you turn in:

```
Random r = new Random(42);
```

For this assignment you are limited to the language features in Chapters 1-5 shown in lecture or the textbook.

Use whitespace and indentation properly. Limit lines to 100 characters. Give meaningful names to methods and variables, and follow Java's naming standards. Localize variables whenever possible. You should construct only one Scanner object for console input. Include a comment at the beginning of your program with basic description information and a comment at the start of each method. Some students try to achieve repetition without properly using while loops, by writing a method that calls itself; this is not appropriate on this assignment and will result in a deduction in points

Your program should be stored in a file called Bagels.java.

Log of execution (user input bold and underlined)

```
Welcome to CSE 143x Bagels!
I'll think up a number for you to guess.
Each digit will be between 1 and 9.
Try to guess my number, and I'll say "fermi"
for each digit you get right and in the right
place, "pico" for each digit you get right
that is in the wrong place, and "bagels"
if you don't get any digits right at all.

How many digits this time? 3
Your guess? 123
pico
Your guess? 456
pico pico
Your guess? 265
pico pico
Your guess? 542
You got it right in 4 guesses
Do you want to play again? y

How many digits this time? 3
Your guess? 123
fermi fermi
Your guess? 456
bagels
Your guess? 789
bagels
Your guess? 122
fermi
Your guess? 113
fermi fermi
Your guess? 133
You got it right in 6 guesses
Do you want to play again? n

Overall results:
    total games   = 2
    total guesses = 10
    guesses/game  = 5.0
    best game     = 4
```