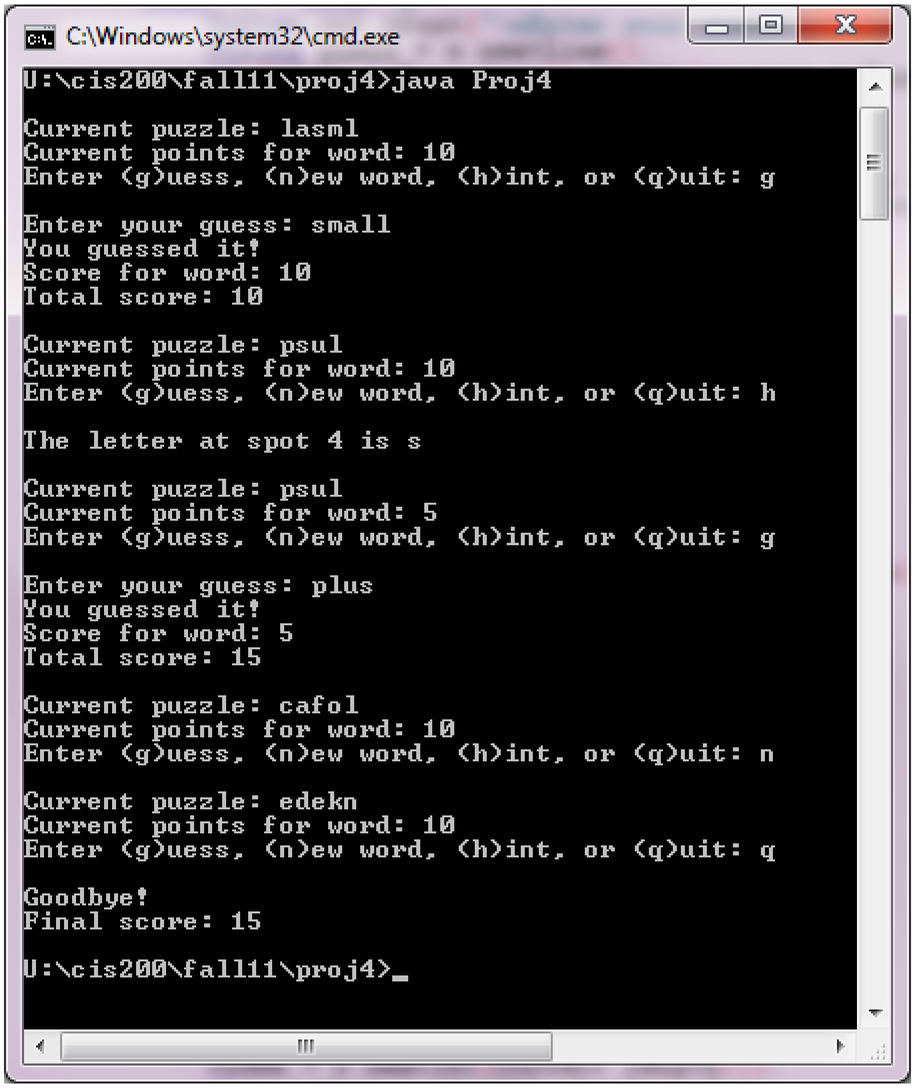
Project 4 (50 points)
Due Friday, October 7 by midnight

Assignment Description:
In this assignment, you will write a jumble game. You will repeatedly pick a random word, randomly jumble it, and let the user guess what it is. The user will also be able to get hints in case they have trouble figuring out the word. Here is an example run of the program:



```
U:\cis200\fall11\proj4>java Proj4

Current puzzle: lasml
Current points for word: 10
Enter (g)uess, (n)ew word, (h)int, or (q)uit: g

Enter your guess: small
You guessed it!
Score for word: 10
Total score: 10

Current puzzle: psul
Current points for word: 10
Enter (g)uess, (n)ew word, (h)int, or (q)uit: h

The letter at spot 4 is s

Current puzzle: psul
Current points for word: 5
Enter (g)uess, (n)ew word, (h)int, or (q)uit: g

Enter your guess: plus
You guessed it!
Score for word: 5
Total score: 15

Current puzzle: cafol
Current points for word: 10
Enter (g)uess, (n)ew word, (h)int, or (q)uit: n

Current puzzle: edekn
Current points for word: 10
Enter (g)uess, (n)ew word, (h)int, or (q)uit: q

Goodbye!
Final score: 15

U:\cis200\fall11\proj4>_
```

Input File:
There is an input file, `words.txt`, available for download from K-State Online. This file has a long list of 4- and 5-letter words. (The first line of the file is the number of words.) Your program should read in all the words in the `words.txt` file as if the file were in the same working directory as your code. I recommend storing these words in a string array.

Requirements:
This program should contain a single class (called `Proj4`) with a `main` method. Your program must compile (by command-line) with the statement:

```
javac Proj4.java
```

It must then run with the command:

```
java Proj4
```

Picking/Jumbling a Word:
You should randomly pick a new word for your game at the following times:

- When the program starts
- After the user has correctly guessed the previous word
- If the user indicates that they want a new word

I recommend using a random number generator (see Project 3) to randomly pick an index in your array of words (which you read from the input file).

Once you have picked the current word, you will need to randomly jumble its letters. We will discuss one technique for jumbling a word during class, but any process that randomly mixes up the letters is fine. You will want to use a random number generator for this step as well.

Keep in mind that it is possible to randomly jumble the word and end up with the original word. This is fine, but it shouldn't happen very often. Also, you may find that several words in the input file contain the same letters. (For example, *meat* and *team*.) If the original word that your program picked was *meat*, and the user guesses *team*, then it is fine if your program says that the guess was incorrect. (It is also fine if you detect this occurrence and give credit to *team* since it uses all the letters and is also a word in the file.)

Giving a Hint:
If the user selects that they want a hint on the current word, then you should randomly pick one of the positions in the "correct" word for the round, and give the user the character at that position. You will want to use a random number generator.

If the user gets several hints on the same word, your program may end up giving them the same hint more than once (since the character chosen is random). That's OK, but if you want to modify your program to give them a different hint each time, then that's fine too.

Keeping Score:
Your program should keep track of the user's total score and their score for the current word. The user should start with 10 possible points for each new word. Every time they guess incorrectly, they should lose a point (unless their possible points are already at 0, in which case

they should stay at 0). Every time they get a hint, the number of possible points for that word should be divided by two (using integer division). If the user guesses the word correctly at some point, the remaining possible points for that word should be added to the total score.

For example, if the user makes two wrong guesses, then needs a hint, and then guesses the word correctly, then they should get 4 points for that word added to their total score. (They lose two points for the two incorrect guesses, and then the remaining 8 points are divided by two when the user needs a hint.)

Documentation:
At the top of the class, add the following comment block:

```
/**
 * (description of the project)
 *
 * @author (your name)
 * @version (which number project this is)
 */
```

Submission:
To submit your project, first create a folder called `proj4`, and move your `Proj4.java` and `words.txt` files into that folder. Then, right-click on that folder and select "`Send To->Compressed (zipped) folder`". This will create the file `proj4.zip`.

Go to "`Files and Content->Modules->File Dropbox`" on K-State Online. Select your lab time and upload the `proj4.zip` file. **Put your name and Project 4 in the description box.**

Grading:
Programs that do not compile will receive a grade of 0. Programs that do compile will be graded according to the following point breakdown:

Requirement	**Points**
Correctly reads words.txt input file	**10**
Repeatedly gets user option (guess, new word, hint, or quit)	**5**
Randomly picks and jumbles word at game start, after a correct guess, and if the user wants a new word	**15**
Guess option works correctly	**5**
Hint option works correctly	**5**
Quit option works correctly	**2**
Score is kept correctly (overall and for the current word)	**5**
Program output exactly matches screenshot	**2**
Documentation/naming/submission	**1**
Total	**50**