Exercise Preparation:
1. Start a terminal application and follow the next steps by typing the appropriate commands:
   a) Create the hw4 directory inside your cs170 directory
      
      mkdir ~/cs170/hw4
   
   b) Copy the files you will need
      
      cp ~/cs170002/share/hw4/* ~/cs170/hw4
   
   c) Move into the hw4 directory (ie make it the working directory)
      
      cd ~/cs170/hw4
   
   d) List the contents of the current directory
      
      ls
   
   e) You should see two files: Blender.java, and TextProc.java

2. Open the Blender.java and TextProc.java file using gedit.

3. There are exercises explained in the comments in both files. Read them carefully and write the code to complete each one. I highly recommend you compile and test frequently.

4. Scoring:
   Blender.java: 40pts
   TextProc.java: 50pts

   Style points: 10 pts. This includes things like making your code neat and readable (indentation and spacing are uniform) and commenting your code appropriately.

Honor Code: Like all work for this class, the Emory Honor Code applies. You should do your own work on all problems, unless you are explicitly instructed otherwise. If you get stuck or have questions, ask your instructor or a TA for help. For EACH of the files you submit, be sure to put the appropriate honor code statement (as specified on the course syllabus) at the top of the file in comments.

Submission:
When you are done, turn-in both files using this commands:

/home/cs170002/turnin-hw Blender.java hw4a
/home/cs170002/turnin-hw TextProc.java hw4b

* You will see a message telling you that your file was successfully submitted along with a date and time. Remember that it is your responsibility to make sure your work is submitted correctly. We DO NOT accept work submitted via email.
Problem decomposition for Blender.java

The purpose of this programming assignment is to practice String methods, loops, and if statements. The program reads a word or phrase from the COMMAND LINE (see Lab 4) and then mixes the letters to form another string. The way it mixes them is by creating a new string that has the first letter of the original string, then the last letter, then the second letter, then the second to last letter, etc. The resulting string has the same length as the original string. The following are a few sample runs:

$java Blender Computer
Croemptu

$java Blender Pei
Pie

Hints:
1) Do an example “by hand”. Notice you will be needing a loop. How many times should the loop run? How would it run differently each time it runs?
2) Also, do an example with a word that has an even number of letters and another one with an odd number of letters. How are these examples different? How can you make your program work for each case?

Problem decomposition for TextProc.java

The purpose of this program is to grade the happiness and the enthusiasm of a given phrase. The user enters a phrase using a Scanner, and the program looks for smiley and sad faces. A int variable happiness, initialized to 0, will increase in 1 each time there is a smiley face, and will decrease in 1 each time it finds a sad face. Similarly, an enthusiasm variable, initialized to 0, will increase in 1 each time there is an exclamation mark (!), and will increase in 3 if the whole message is written in capital letters.

A smiley face is either :) or =)
A sad face is either :( or =( 

The following are sample runs:

$java TextProc
Please enter a phrase
I am very disappointed :( :( 

The happiness of this phrase is: -2
The enthusiasm is: 0

$java TextProc
Please enter a phrase
I am so happy!!! :) 

The happiness of this phrase is: 1
The enthusiasm is: 3
$java TextProc
Please enter a phrase
THIS EXAMPLE IS GREAT!! =) =)
The happiness of this phrase is: 2
The enthusiasm is: 5

Hints:
1) Checking for exclamation marks is easier than smiley or sad faces. For checking for the smiley and sad faces, follow the example for the exclamation marks.
2) In the case of smiley or sad faces, look character by character (using a loop) on the phrase. If the current position and the next one is what you are looking for, then add one to happiness. Be careful not to go beyond the limits of the string, or you will have a StringIndexOutOfBoundsException.
3) For checking if the whole phrase is in capital letters, again check character by character. This problem is similar to the one in Homework 3, but this needs a loop to check all the characters instead of just one. Also, you will need to add a condition to the if statement, so that it doesn't consider spaces when you're looking for capital letters. Use the “flag technique” to check for all capital letters.