Programs using String Methods

Exercise 1 – SeparateLine

Write a program that gets a String from the user and then prints the characters each on a separate line.
 	
Hint: Use the length() method and a loop.
	Sample Output
	
	`Enter a word: hello`
	`h`
	`e`
	`l`
	`l`
	`o`

Exercise 2 – ReverseOrder

Write a program that gets a String from the user and then prints the characters in reverse order.	Sample Output
	
	`Enter a word:  hello`
	`olleh`

Exercise 3 – AddressBook

Write a program that prompts the user for two names. Then compare them to determine which name should be listed first alphabetically in an address book.
 	
Use both `.equals()` and `.compareTo()` methods when comparing the string.
	Sample Output
	
	`Enter a name:  Thomas`
	`Enter another name:  Edward`
	
	`Edward first then Thomas.`
	`Thomas is not the same name as Edward.`

Exercise 4 – RomanNumeral

Write a program that gets a Roman Numeral from the user and then converts it to its decimal equivalent, where **I = 1** and **V = 5** and **X = 10**.
 	
Assume that the user will not enter a number higher than X.
	Sample Output
	
	`Enter a Roman Numeral (I to X):  vii`
	`Here is the decimal conversion:  7`

Exercise 5 – CountVowels

Create a CountVowels application that prompts the user for a string and then displays a count of the number of vowels in the string.	Sample Output
	
	`Enter text: Java programming is fun`
	`The number of vowels in Java programming is fun is 7`

Exercise 6 – DayOfWeek (*See Chapter 2, R2.19*)

Write a program that prints the day of the week as a string:
 	
`Sun Mon Tue Wed Thu Fri Sat`
 	
given the day of the week as a number (1 – 7) as input.
 	
Hint: Store the day abbreviations as one string; then find a substring by multiplying the input by 3
	Sample Output
	
	`Enter a number for the day of a week (1-7): 5`
	`Day of the week is Thu`

Exercise 7 – LetterSwap (*See Chapter 2, R2.20*)

Write a program that swaps two letters in a word. The user will enter a word and the two positions of the letters to swap.
 	
Hint: Create a first substring to store the first part of the word before the first letter. Create a middle substring to store the middle part of the word after the first letter and before the second letter. Create a last substring to store the last part of the word after the second letter. Join the word together using five substrings.
	Sample Output
	
	`Enter a word: football`
	`Enter two positions to swap (1 - 8):`
	`4`
	`7`
	`foolbatl`

Exercise 8 – WordGuessGame

Write a program that allows a user to guess the letters of a secret word. Output a word using hyphens to represent the length of the word. Repeatedly, prompt the user for a letter guess and check the letter for a match. If a letter guess matches a letter in the word, replace the corresponding dash with the letter. Letters may be entered as uppercase or lowercase, but only uppercase letters should be displayed.
 	
Hint: Assign dashes (the length of the secret word) to a variable "`wordSoFar`". Get a letter from the user. Iterate through each letter of the secret word. For each iteration, check if the letter guess matches a letter. If true, assign to a new variable "`newWord`" a substring of `wordSoFar` (from 0 to letter match). Join the letter guess to `newWord`. Join the remainder of the `wordSoFar` (from letter match to the end). Update `wordSoFar` with `newWord`.
	Sample Output
	
	`The computer is "thinking" of a secret word . . .`
	`Here is the word as a set of dashes.`
	`------`
	
	`Enter a letter: b`
	`B-----`
	
	`Enter a letter: e`
	`B-----`
	
	`Enter a letter: n`
	`B-N-N-`
	
	`Enter a letter: a`
	`BANANA`
	
	`You guessed the word!!`
	`The secret word is BANANA`