**Winter Break Challenge Programs - Ideas and Resources**

**Program # 1 - Palindrumbers**

* Page 347 in textbox has syntax for checking for String palindromes
* Use *break* statements appropriately to make your program cut out needless calculations
* Nested *for* loops or *while* loops are a good idea. You may need three levels of loops since you have to check for the product of three 3-digit numbers.
* The (length % 2) will tell you if you have an even or odd digit number.
* The *valueOf* String method can convert a String to its integer equivalent. (Page 344)
* The Integer wrapper class has the *parseInt*() function which converts a String to its integer equivalent. (Page 394-395 shows examples)

**Program # 2 - SpecialPrimes**

* You will not need to use any value greater than 10 for a and b, which should help your program minimize its looping and overflow issues.
* You may choose to create an array or ArrayList of all prime numbers to be used to compare against the 5 digit numbers generated. Then again, you may do it a different way.
* Remember that num1 is a factor of num2 if: num2 % num1 == 0. (no remainder)

**Program # 3 - Slurbitongs**

* One solution requires just 4 methods:
	+ toOctal - Converts from Base 10 to Base 8 (divide by powers of 8)
	+ toDecimal - Converts from Base 8 to Base 10 (multiply by the power of 8)
	+ sort() - Sorts the digits into increasing (ascending) order, dropping off all leading zeros.
	+ isSorted() - Boolean method that determines if the number is already sorted
* Pages 348 to 350 in your book show Decimal to hexadecimal conversion (Base 16)
* Remember that octal numbers only use the symbols 0 through 7.
* The Integer wrapper class can make your conversions MUCH easier, if you research it.

**Program # 4- Substitution**

* You can assume that there will be at most four different letters in the equation.
* You will need to replace each letter with all possible values to check for all possible solutions.
* You will definitely have some type conversions in this program, with the following very likely:
	+ String to character Array
	+ character array to String
	+ String to integer
* Nested for loops (or while statements) are likely a necessity.
* Look into your *replaceAll* String method for swapping out all occurrences of a letter with the same integer.

ArrayLists are a wonderful container object since we don't have to declare a size, we just add to it and it creates a new index each time we add a new object. However, we can't add primitive data types to them, so we would have to use the wrapper class Integer, in place of int. Look at pages 430-435 for more information on ArrayLists.