**AP Computer Science - Winter Break Challenge Programs**

**Program # 1 - Palindrumber**

A palindrome is a word which is the same when spelled backwards. For example, the word racecar is a palindrome. Similarly, a palindrumber is a number which is the same when its digits are arranged backwards. (Ex: 92029) The smallest palindrumber made from the product of two 2-digit positive integers that has an even number of digits is 1001. (91 \* 11 = 1001) The smallest palindrumber made from the product of two 2-digit positive integers that has an odd number of digits is 121. (11 \* 11 = 121)

Write an java program to find the smallest palindrumber made from the product of three 3-digit positive integers that have even and odd numbers of digits. The program must calculate this value, not just report it.

The smallest palindrumber made from the product of three 3-digit positive integers with an even number of digits is <answer>.

The smallest palindrumber made from the product of three 3-digit positive integers with an odd number of digits is <answer>.

**Program # 2 - SpecialPrimes**

This problem will focus on all 5 digit integers (10,000 to 99,999) that can be written in the form:

*xayb* where *x* and *y* are two different prime numbers and x and y are the **only** prime factors of the 5 digit number being evaluated. You can assume *a* and *b* are non-negative integers, but obviously they will not be prime. The goal of your program is to calculate how many times does a given digit, *d,* appear in all 5 digit numbers that satisfy the above conditions. (where *d* is 0 to 9) The user will supply x, y and d in the program.

This example finds all 5 digit numbers whose only prime factors are 13 and 19. It also outputs the total number of 0s that appear)

13 19 0 <User Input>

41743

61009

89167

The value 0 appears 2 times in all special primes of 13 and 19.

**Program # 3 - Slurbitongs**

The slurbitong value of a number x is found by following these five steps:

1. First, convert x to base 8. Call this p.
2. Next, sort the digits of p in increasing order. Call this q.
3. Subtract q from p (in base 8, of course).
4. Repeat the sort-subtract sequence 4 more times, or until the digits in the result are in sorted order (whichever comes first).
5. Finally, convert the number back to base 10.

Example: 3418 has a slurbitong value of 1008. Here are the steps:

1. 3418 = 65328; p = 6532
2. 6532 sorted into increasing order means: q = 2356
3. Subtract q from p (in base 8) 65328- 23568 = 41548; Since digits are not in sorted order,
4. Repeat sort subtract (Steps 2 & 3) 4 more times or until the digits are in sorted order, whichever comes first:
   1. 4154 - 1445 = 2507 (still not sorted) (Repeat # 1)
   2. 2507 - 257 = 2230 (still not sorted) (Repeat # 2)
   3. 2230 - 223 = 2005 (still not sorted) (Repeat # 3)
   4. 2005 - 25 = 1760 (still not sorted) (Repeat # 4, last repeat)
5. Convert 1760 back to base 10. 1760 = 1008

Each input will be a positive integer less than 1,000,000. Your output should only be the slurbitong value of each.

3418 <user input>

1008 <expected output>

**Program # 4 - Substitution**

Consider the equation: ABCD + BDC = EAEA

The goal of your program is to generate possible values for all variables in the equation and provide all possible solutions.

The replacements must be consistent; that is, replace all A's with the same digit, all B's with another digit, and so on.

A solution to the above example is A =1, B = 5, C=3, D=8 and E=2. Thus, the equation would read: 1538+583 = 2121

There are 5 more solutions: 1547+574 = 2121, 1574+547 = 2121, 1583 + 538 = 2121, 3658 + 685 = 4343, and 3685 + 658 = 4343.

User input consists of three strings, x, y, and z, where x + y = z and each is a string of characters. The strings can be composed of any letters A through Z or numbers. Any numbers are treated as their decimal equivalents while the letter consistently represent a specific digit. Your output needs to report the number of solutions as well as outputting those solutions.

To keep the run time reasonable, you can assume that each equation will contain no more than 4 different letters.

Example:

XYZ + XY = 6QQ <User Input>

There are 3 possible solutions

546 + 54 = 600

576 + 57 = 633

586 + 58 = 644