A+ Practice Site - Advanced String Problems

Problem Types:

- **Reverse Words**
- **Palindromes and Least Palindromes**
- **Removing/Replacing Strings**
- **Shifting/Reversing Letters**
- **Traversing Strings and Matrices**

On the following pages, there are ten practice problems. The dat files and solutions to these problems are provided in a separate folder.

Problem	**Key Concepts**
pr70 - Reverse	remove characters from string; traverse from both beginning and end of string
pr71 - Palindrome	remove characters from string; traverse from both beginning and end of string
pr72 - Least Palindrome	`StringBuffer`; `insert` and `charAt`; helper method
pr73 - HTML	`asList` to put array into `ArrayList`; `indexOf`; `delete`; do until done; don't print blank lines
pr74 - Encrypted Words	`StringBuffer`; `reverse`; `substring`; cases
pr75 - Word Stats	`indexOf`; running sums; averages; `charAt`; `printf` to right justify and round
pr76 - It's Justified	`StringBuffer`; `insert`; `lastIndexOf`; `indexOf`; counting characters
pr77 - Alpha-split	linear search; concatenate char to strings
pr78 - The Matrix	read matrix with `toCharArray`; traverse a matrix; `&&` statement in for loop; read forwards and backwards; writing methods for cases; check for staying within matrix
pr79 - Censored!	sort by length of word; `StringBuffer` to `String`; `lowercase`; repeated words; replace with `substring`;

pr70 - Reverse

Problem: Write a program that print out PALINDROME backwards 10 times.

Input: none

data file: `none`

Output: Print out the word PALINDROME backwards 10 times.

Assumptions: A loop may make this problem easier.

Sample Input:
```
none
```

Sample Output:
```
EMORDNILAP
EMORDNILAP
EMORDNILAP
EMORDNILAP
EMORDNILAP
EMORDNILAP
EMORDNILAP
EMORDNILAP
EMORDNILAP
EMORDNILAP
```

pr71 - Palindrome

Problem: Write a program that will determine if a string is a palindrome when all non-alphabetic characters are removed from the string.

Input: The first line of the data set is an integer that represents the number of lines that follow. Each of the remaining lines contains a list of words. Strings will contain only uppercase letters of the alphabet.

data file: `pr71.dat`

Output: PALINDROME if the string is a palindrome and NOT PALINDROME is it is not.

Assumptions: A string is a palindrome if the letters in the string are the same when read forwards or backwards.

Sample Input:

```
4
RACE CAR
GO HANG A SALAMI, I'M A LASAGNA HOG!
ALAN ALDA
A TOYOTA
```

Sample Output:

```
PALINDROME
PALINDROME
NOT PALINDROME
PALINDROME
```

pr72 - Least Palindrome

Problem: Write a program that will determine the least palindrome for a given string. A least palindrome is the palindrome that is constructed using the least number of letters possible to make a word become a palindrome. All letters will be uppercase.

Input: The first line of the data set is an integer that represents the number of lines that follow. Each of the remaining lines contains a word. Each word will contain only uppercase letters of the alphabet.

data file: `pr72.dat`

Output: Output the least palindrome for each word.

Assumptions: A string is a palindrome if the letters in the string are the same when read forwards or backwards.

Sample Input:

```
4
COMPUTER
SCIENCE
BANANA
RACECAR
```

Sample Output:

```
COMPUTERETUPMOC
SCIENCECNEICS
BANANAB
RACECAR
```

pr73 - HTML

Problem: Write a program that will remove all the HTML tags from a markup language program that is used to create web pages. In a program, the programmer would use <b> some text </b> to make some text appear in bold type. Only the following tags may appear in the program:

<html>, </html>, <body>, </body>, <title>, </title>, <head>, </head>, <b>, </b>, <table>, </table>, <tr>, </tr>, <td>, </td><br>.

Input: The input consists of an unknown number of code lines with HTML tags. All tags will be lower case.

data file: `pr73.dat`

Output: Output the lines of code with the HTML tags removed. Blank lines should be removed.

Assumptions: None.

Sample Input:
```
<html><head><title><b>Java is fun</b></head>
</title>
<body>
This text is to remain
3 < 5 and 7>4 are both true.<br>
17 % 4 <= 2 is false. So is 9 > 10 <br>
<b<html>>
<b>Java is fun!</b>
</body>
</html>
```

Sample Output:
```
Java is fun
This text is to remain
3 < 5 and 7>4 are both true.
17 % 4 <= 2 is false. So is 9 > 10
Java is fun!
```

pr74 - Encrypted Words

Problem: Write a program that will encrypt the words of a sentence as follows:

For the words in the odd numbered positions (i.e. the 1st, 3rd, 5th, ...) of the sentence, you will reverse the letters the words - so COMPUTER would become RETUPMOC.

For the words in the even numbered positions (i.e. the 2nd, 4th, 6th, ...) of the sentence, you will have their letters rotated in a cyclic pattern - so `SCIENCE` would become `CESCIEN` if it were the second word in the sentence or `ENCESCI` if it were the fourth word in the sentence.

If the position of the word is greater than the length of the word, the word is printed without change.

Input: The first line of the data set is an integer that represents the number of lines that follow. Each of the remaining lines contains a sentence of uppercase letters of the alphabet.

data file: `pr74.dat`

Output: Output the original sentence on one line and the encrypted sentence on the next. Place at least one blank line between test cases.

Assumptions: None.

Sample Input:

```
3
PETER PIPER PICKED A PECK OF PICKLED PEPPERS
WITHOUT COMPUTERS A GEEK WOULD HAVE NOWHERE TO HIDE
ENCRYPED WORDS IS A FUN PROGRAM THAT STRETCHES YOUR IMAGINATION
```

Sample Output:

```
RETEP ERPIP DEKCIP A KCEP OF DELKCIP SPEPPER
TUOHTIW RSCOMPUTE A GEEK DLUOW HAVE EREHWON TO EDIH
DEPYRCNE DSWOR SI A NUF ROGRAMP TAHT TRETCHESS RUOY MAGINATIONI
```

pr75 - Word Stats

Problem: Write a program that will provide statistics to the author of a word processing paper. The statistics needed are: the number of vowels, the number of consonants, the number of spaces, the number of other characters that are non-alphabetic and not a space, the number of words, the length of the longest word, the length of the shortest word, and the average length of a word rounded to tenths.

Input: A paragraph of text.

data file: `pr75.dat`

Output: Output the statistics in the format given below. All numbers are right justified under their heading.

Assumptions: The paper is one paragraph long. A word is one or more characters surrounded by a space. There will not be two consecutive spaces in the text.

Sample Input:
```
/* This is a comment */ 2 b || ! 2 b, that is the question. Now is the time for all
good men to come to the aid of their country. What do you think of this programming
problem?
```

Sample Output:
```
VOWELS - 46
CONSONANTS - 78
SPACES - 39
OTHER - 13
WORDS - 40
LONGEST - 11
SHORTEST - 1
AVERAGE - 3.4
```

pr76 - It's Justified

Problem: Write a program that will print a paragraph in forty (40) columns fill justified. Fill justified means that text is lined up on the left side and the right side of the 40 columns. To fill justify, an extra space is placed after words, beginning with the left most word, and continuing as long as necessary so the 40 characters are filled. The last line will be left justified without any extra spaces.

Input: A paragraph of text.

data file: `pr76.dat`

Output: Output the column numbers as shown below. Then output the paragraph fill justified with 40 characters per line.

Assumptions: None.

Sample Input:

To fill justify, an extra space is placed after words, beginning with the left most word, and continuing as long as necessary so the 40 characters are filled. The last line will be left justified without any extra spaces.

Sample Output:

```
1234567890123456789012345678901234567890
To  fill  justify,  an  extra  space  is
placed  after  words, beginning with the
left  most  word, and continuing as long
as  necessary  so  the 40 characters are
filled.  The  last  line  will  be  left
justified without any extra spaces.
```

pr77 - Alpha-split

Problem: Write a program that will print the letters of a string so the letters in the range A .. M are printed on the first line and the letters in the range N .. Z are printed on the second line.

Input: The first line of the data set is an integer that represents the number of lines that follow. Each of the remaining lines contains a sentence in uppercase letters of the alphabet. Each word is separated by one space.

data file: `pr77.dat`

Output: Output the letters of each string so the letters in the range A .. M are printed on the first line and the letters in the range N .. Z are printed on the second line. Non-alphabetic characters are to be printed on both rows. Print at least one blank line between data sets.

Assumptions: None.

Sample Input:
```
2
TODAY IS DECEMBER 25, THE HAPPIEST DAY OF THE YEAR!
NOW IS THE TIME FOR ALL GOOD MEN TO COME TO THE AID OF THEIR COUNTRY.
```

Sample Output:
```
  DA  I  DECEMBE  25,  HE HA  IE   DA   F  HE  EA !
TO  Y  S        R 25, T     PP  ST   Y O  T   Y  R!

    I   HE  IME F   ALL G  D ME      C ME      HE AID  F  HEI  C      .
NOW  S T   T     OR       OO    N TO  O   TO T       O  T   R  OUNTRY.
```

pr78 - The Matrix

Problem: Write a program that will determine if a word appears in a matrix of characters. The word may appear vertically, horizontally, or diagonally and either forwards or backwards.

Input: The matrix is square and contains no more than 20 characters. Following the matrix is a list of words.

data file: `pr78.dat`

Output: For each word that follows the matrix, output the word followed by `APPEARS IN THE MATRIX` or the word followed by `DOES NOT APPEAR IN THE MATRIX`.

Assumptions: All letters are uppercase.

Sample Input:
```
APPLEXY
PXLHJKE
EDEGGLL
XXCGFPD
GOGNMYN
TAHUUPU
QDGBTSB
APPLE
AXE
APEX
CAT
HEX
CAR
HAT
COMPUTER
GUM
BUNDLE
TUG
ELF
```

Sample Output:
```
APPLE APPEARS IN THE MATRIX
AXE APPEARS IN THE MATRIX
APEX APPEARS IN THE MATRIX
CAT DOES NOT APPEAR IN THE MATRIX
HEX APPEARS IN THE MATRIX
CAR DOES NOT APPEAR IN THE MATRIX
HAT APPEARS IN THE MATRIX
COMPUTER DOES NOT APPEAR IN THE MATRIX
GUM APPEARS IN THE MATRIX
BUNDLE APPEARS IN THE MATRIX
TUG APPEARS IN THE MATRIX
ELF APPEARS IN THE MATRIX
```

pr79 - Censored!

Problem: Write a program that censors certain words from a line of text. The letters in the censored words are to be replaced by asterisks (`*`). The censored word may appear alone or may be contained in a word. If a censored word is contained in a censored word, the longest word should be removed. If a line contains more than one censored word, all instances should be removed.

Input: The first line of the data set is a list of words to censor. Each word is separated by a single space. On each of the following lines, there is a sentence. Letters may be uppercase or lowercase.

data file: `pr79.dat`

Output: Output each sentence with the censored words replaced with asterisks.

Assumptions: Censored words will not be over lapping (i.e. the end of one word is not the beginning of another) even though a censored word could be entirely contained in a censored word.

Sample Input:
```
the at rich poor chocolate ice
The richest people in the world are chocolate lovers.
Poor Richard's Almanac is the best choice for getting sage advice.
Rich chocolate and licorice threaten the rich and poor alike.
```

Sample Output:
```
*** ****est people in *** world are ********* lovers.
**** ****ard's Almanac is *** best cho*** for getting sage adv***.
**** ********* and licor*** thre**en *** **** and **** alike.
```