Programming With Characters

Caesar cipher

Caesar cipher is a simple technique of encryption of plain text by replacing every character in the plain text by a character fixed number of positions down the list of the alphabet. The last characters are folded back to the beginning.

Shift: 5	
Original	Encrypted
'A'	'F'
'B'	'G'
...	...
'V'	'A'
'W'	'B'
'X'	'C'
'Y'	'D'
'Z'	'E'

Caesar cipher

Write a C program that will read a text stream (either from stdin or from some redirected text file) and will encrypt the English alphabet, 'A', ..., 'Z', 'a', ..., 'z', using Caesar cipher. The value of shift should be within $1 - 10$ and will be decided by the rand() function.

Caesar cipher

```
/*
 * caesar.c
 */
#include <stdio.h>
#include <stdlib.h>    // rand(), srand()
#include <ctype.h>     // isalpha()
#include <sys/types.h> // for getpid()
#include <unistd.h>    // for getpid()
int main()
{
    char c, shift ;
```

```
    srand(getpid()) ;
    shift = (char)(rand()%10 + 1) ; // Generating shift

    while((c = getchar()) != EOF){
        if(isalpha(c)) {
            if(isupper(c)) putchar((c-'A'+shift)%26+'A');
            else putchar((c-'a'+shift)%26+'a');
        }
        else putchar(c) ;
    }
    putchar('\n') ;
    return 0 ;
}
```

Character, Word and Line Count

Write a C program that reads a text file and counts the total number of characters, including the non-printable (invisible) characters like white spaces (`' '`, `'\n'`, `'\t'`); the total number of words and the total number of lines in the text.

Words & Lines

- A sequence of non-whitespace characters separated by one or more whitespaces. The following text has 10 words.

  "`* What can be said at all,can be said clearly*`"

  '*', "all,can" and "clearly*" are counted as single words.

- Every newline character ('\n') defines a line.

Steps to Follow

1. Three counters are initialized to zero: charCount, wordCount and lineCount.
2. Characters are read upto the end-of-file (Ctrl-D for the keyboard). charCount is incremented each time a character is read.
3. If there is a transition from a whitespace to a non-whitespace character, the wordCount is incremented. It is necessary to save the

previous character in a variable to detect the transition.

4. The lineCount is incremented if the newline character ('\n') is read.
5. Finally the content of the three counters are printed.

```
/*
 * cwlCount.c
 */
#include <stdio.h>
int main()
{
    int charCount = 0, wordCount = 0;
    int lineCount  = 0 ;
    char prevChar = ' ', newChar ;

    printf("Enter the text\n") ;
    while( (newChar = getchar()) != EOF) {
        switch(newChar) {
                case  ' '  :
```

```
                case  '\t' : break ;
                case  '\n' : ++lineCount ;
                             break ;
                default :
                     if(prevChar == ' ' ||
                        prevChar == '\t' ||
                        prevChar == '\n') ++wordCount ;
            }
            ++charCount ; prevChar = newChar ;
    }
    printf("Lines = %d\nWords = %d\nChars = %d\n",
         lineCount, wordCount, charCount) ;
    return 0 ;
}
```

Redirecting a *text* (ASCII) File

```
$ ./a.out < t.c
Enter the text
Lines = 26
Words = 96
Chars = 655
$
```