28 Feb 2023

COMP281 2022-2023 – Assignment 2

- In the following, you will find the problems that constitute Assignment 2. They will be also available via Code Grade on the canvas module for COMP281.

- You need to write a C program (not C++ or C#) that solves each problem – it must read the input, as specified in the problem description then print the solution to the given problem for that input.
    - Note that code is "correct" only if it correctly implements a solution to the problem stated in the assignment, not "if Code Grade accepts it".
    - That is, even if Code Grade accepts your code, it could be wrong. Read the problems carefully.

- Input is read from the standard input, in the same way that you read input from the keyboard as shown in lectures (e.g., using scanf). Output is also printed to the standard output, as you have seen (e.g., using printf).

- You must also include a brief report describing your solutions to the problems. This should be the equivalent of maximum two sides of A4 paper and should give a description of how each of your solutions works. This should include describing the algorithm used to reach the solution, describing your use of any C language features (that were not discussed in lectures) and identifying any resources that you have used to help you solve the problems. A separate canvas assignment called "Assignment 2 brief report" has been setup for this.

- This assignment is worth 50% of the total mark for COMP281.
    - Both problems are weighted equally.
    - For each problem, your solution can earn a total of 50 points (scaled from 20 points per solution in assignment 1).
        - 25 points for "Functionality and Correctness" awarded for programs that correctly solve the problem for all test cases.
        - 20 points for "Programming style, use of comments, indentation and identifiers" awarded depending on the style, comments and efficiency of the solution
        - 5 points for the quality and depth of the accompanying report
    - The final grade results from normalising the earned points to a scale of 100.
    - See separate "comp281-detailed-marking-guidelines.pdf" for more details (note that in that document, the mark deductions referred to will be scaled by 2.5 for assignment two, since the document refers to assignment 1, where there were 5 parts vs the 2 parts to this assignment. e.g. if an issue for a program in assignment one had a penalty of up to four marks, the same issue found in a program for assignment two will have up to a ten mark penalty).

Submission Instructions

- Submit your solution to each part of the assignment via Canvas and don’t forget to include your brief report.
- The deadline for this assignment submission is **Monday 13-Mar-2023 at 12:00 (midday)**
- Penalties for late submission apply in accordance with departmental policy as set out in the student handbook, which can be found at: http://intranet.csc.liv.ac.uk/student/ug-handbook.pdf

Part 1.

Title: Run Length Encoding of ASCII Art

Description

In the very early days of computing, images were produced using ASCII characters. Such ASCII Art was very popular and, since storage was limited, simple techniques were developed to compress such images. One such simple technique of lossless compression was Run Length Encoding. This scans an input image for runs (sequences) of 3 or more identical values, and replaces them with three such characters followed by an INT string representing a count of the number of characters, and terminated by a * e.g.

A line from a piece of ASCII Art image as follows:

```
,,,,,,]F                                                8,
```

which is composed of 6 commas, a closed square bracket, a capital F, 48 spaces an eight and a comma, would be represented as:

```
,,,6*]F   48*8,
```

Write a program assignment2_1.c to accept as input either the letter "C" or the letter "E" followed by a newline. Each subsequent line of input data until EOF contains image data.

If the initial input was "C" this represents "compress", and the lines of input text represent an ASCII Art image. Process this input to compress it using the rules specified above, and output the compressed version.

If the initial input was "E" this represents "expand", and the lines of input text represent a compressed ASCII Art image. Process this input and expand it using the rules referred to above. Print out this uncompressed ASCII Art image.

You can test your program by compressing an ASCII Art image, saving the output to a file, editing the file to add E as the first line, then feeding this back into your program to effectively reproduce the original ASCII Art image. If your program is working correctly then the original image and your compressed then decompressed version should be identical.

Input

Either an ASCII art file with a C on the first line (this will be compressed using RLE)

or

A compressed ASCII art file with an E on the first line (this will be expanded).

Output

After compressing an input file, all runs of three or more of any character should be replaced by three of the character, an integer count and a star.

After expanding an input file, all previously compressed sequences should be replaced by the original character runs.

Sample Input

```
C
                               #####
                 #######      #**#!!###
                #**#!!!!##   #****#!!!!#
               #****###!!!#  #*****#!!!!#
               #*******#!!!# #******#!!!!#
               #*********#!###!*!*!*#!!!!!#        --
               #!*!*!*!*!*!#!##########!!!!#      /_
               ###########!##!!!!!!!!!!#!!!#     //__
            ###!!!!!!!!!!!#!!!!!!!!!!!!!#!!!####///  \
  \       ##!#!!!!!!!!!!!#!!!!!!!!!!!!!!!#!!!!!!!#
  _\    ##!!#!!!!!!!!!!!#!!!!!!!!!!!######!!!!!!!*#
   \\  ##!!#!!!!!!!!!!!!#!!!!#######     #!!!!!!***#
 ___\\#!!!###################*****       #...!!*****#
/   \#!!!.#       ***** #     ***        #....*******#
    #*....#        ***   #              #.......*****#
   #**.....##          *****          ##........!!****#
   #!........##       *******#########......#...!!!!!*#
  #!...........#######.*****...............#.#..!!!!**#
 #*.....##.............#..#...............#...#.!!****#
 #*....#.#............#....#............##......!*****#
 #*.......##.......###......###........#.......!!!****#
 #*.........#######......!!....########.......!!!!!***#
  #!!!.................!!!!!!!!.............!!!*******#
   #!!!!............!!!!!!!!!!!!!!!!!!!!!!!!!!!******#
    #*******!!!!!!!!!!!!!!!!!!!!!!!!!!!!***!!!!*****#
     #******!!!!!!!!!!!!!!!!!!!!!!!!!********!!****#
      ##*****!!!!!!!!!!!!!!!!!!!!!#*************###
        ##****!!!!!!!!!!!!!!!!!!!!!###******####
          ####!!!!!!!!!!!!!!!!!!!!!!!!######!#
              #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*#
              #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!***##
             #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!******#
            #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*******#
           #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*****#
          #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!**#
         #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!##
        #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*##
       #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!***#
       #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!####!!!!!!****##
      #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!###****##!!!!******##
      #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!##**!!*****#!!!********#
      #!!!!!!!!#!!!!!!!!!!!#!!!!!!!#***!!!!***!!!!**********#
     #!!!!!!!!!!#!!!!!!!!!#!!!!!!!#****!!!!!*!!!!!!!!!!*****#
     #!!!!!!!!!!!#!!!!!!!#!!!!!!!#*!!***!!!!!!!!!!!!!!!!!***#
     #!!!!!!!!!!!!#!!!!!!#!!!!!!#*!!!!!*!!!!!!!!!!!!!!!*****#
     #!!!!!!!!!!!!#!!!!!!#!!!!!#***!!!!!!!!!!!!!!!!!********#
    #!!!!!!!!!!!!!#!!!!!!#!!!!!#****!!!!!!!!!!!!!!**********#
    #!!!!!!!!!!!!!#!!!!!!#!!!!!#*****!!!!!!!!!!!!!!*********#
    #!!!!!!!!!!!!!#!!!!!!#!!!!!#***!!!!!!!!!!!!!!!!!********##
   ##!!!!!!!!!!!!!#!!!!!!#!!!!!#*!!!!!!!!!!!!!!!!!!!!!*****#!*##
  #!#!!!!!!!!######!!!!!!#!!!!!#**!!!!!!!!!!#########!!!!*#!!**##
 #!#!#!!!!!!#!!!!!!!!!!!!#!!!!!#***!!!!!####******!!!#######!!**#
#!#!!##!!!!#!!!!!!!!!!!############*!!!#********!!!!!!!!!!!!!!!**#
#!#!!#!#!!#!!!#!!!!!!!#!!!!!!!!!!!!!!!#***********!!!!!!!!!!!!!!!#
#!#!!!#!#!#!!#!!!!!!!#!!!!#!!!!#!!!!!#**********!!!!!!!!!!!!!!!**#
#!!#!!!#!##!!#!!!!!!#!!!!#!!!!#!!!!!!#************!!!!!!!!!!****#
 ######### ##########!!!!#!!!!#!!!!!!#**********!!!!!!!!!!!!***#
                     #################************!!!!!!!!!!**#
                                     #**********!!!!!#########
                                      ###############
```

Sample Output

```
   32*###5*
   18*###7*   6*#**#!!###3*
   17*#**#!!!4*##   3*#***4*#!!!4*#
   16*#***4*###3*!!!3*#  #***5*#!!!4*#
   16*#***7*#!!!3*# #***6*#!!!4*#
   16*#***9*#!###3*!*!*!*#!!!5*#   8*--
   16*#!*!*!*!*!*!#!###10*!!!4*#   6*/_
   16*###11*!##!!!10*#!!!3*#   5*//__
   13*###3*!!!11*#!!!13*#!!!3*###4*///3*  \
   3*\   7*##!#!!!11*#!!!15*#!!!7*#
   3*_\   4*##!!#!!!11*#!!!11*###6*!!!7**#
   4*\\  ##!!#!!!12*#!!!4*###7*   5*#!!!6****3*#
  ___3*\\#!!!3*###19****5*   7*#...3*!!***5*#
 /   3*\#!!!3*.#   7****5* #   5****3*   8*#...4****7*#
   5*#*...4*#   8****3*   3*#   14*#...7****5*#
   4*#**...5*##   10****5*   10*##...8:Q
4*#
   4*#!...8*##   7****7*###9*...6*#...3*!!!5**#
   3*#!...11*###7*.***5*...15*#.#..!!!4***#
  #*...5*##...13*#..#...15*#...3*#.!!***4*#
  #*...4*#.#...12*#...4*#...12*##...6*!***5*#
  #*...7*##...7*###3*...6*###3*...8*#...7*!!!3****4*#
  #*...9*###7*...6*!!...4*###8*...7*!!!5****3*#
   3*#!!!3*...17*!!!8*...13*!!!3****7*#
   4*#!!!4*...12*!!!27****6*#
   5*#***7*!!!28****3*!!!4****5*#
   6*#***6*!!!25****8*!!***4*#
   7*##***5*!!!21*#***13*###3*
   9*##***4*!!!21*###3****6*###4*
   11*###4*!!!24*###6*!#
   15*#!!!30**#
   15*#!!!29****3*##
   14*#!!!29****6*#
   13*#!!!29****7*#
   12*#!!!32****5*#
   11*#!!!35***#
   10*#!!!39*##
   9*#!!!41**##
   8*#!!!43****3*#
   8*#!!!33*###4*!!!6****4*##
   7*#!!!31*###3****4*##!!!4****6*##
   7*#!!!29*##**!!***5*#!!!3****8*#
   7*#!!!8*#!!!11*#!!!7*#***3*!!!4****3*!!!4****10*#
   6*#!!!10*#!!!9*#!!!7*#***4*!!!5**!!!10****5*#
   6*#!!!11*#!!!7*#!!!7*#*!!***3*!!!17****3*#
   6*#!!!12*#!!!6*#!!!6*#*!!!5**!!!15****5*#
   6*#!!!12*#!!!6*#!!!5*#***3*!!!17****8*#
   5*#!!!13*#!!!6*#!!!5*#***4*!!!14****10*#
   5*#!!!13*#!!!6*#!!!5*#***5*!!!14****9*#
   5*#!!!13*#!!!6*#!!!5*#***3*!!!17****8*##
   4*##!!!13*#!!!6*#!!!5*#*!!!21****5*#!*##
   3*#!#!!!8*###6*!!!6*#!!!5*#**!!!10*###9*!!!4**#!!**##
  #!#!#!!!6*#!!!12*#!!!5*#***3*!!!5*###4****6*!!!3*###7*!!**#
 #!#!!##!!!4*#!!!11*###12**!!!3*#***8*!!!15***#
 #!#!!#!#!!#!!!3*#!!!7*#!!!15*#***11*!!!15*#
 #!#!!!3*#!#!#!!#!!!7*#!!!4*#!!!4*#!!!5*#***10*!!!15***#
 #!!#!!!3*#!##!!#!!!6*#!!!4*#!!!4*#!!!6*#***12*!!!10****4*#
  ###9* ###10*!!!4*#!!!4*#!!!6*#***10*!!!12****3*#
   22*###17****12*!!!10***#
   38*#***10*!!!5*###9*
   39*###15*
```

Hint

You'll need to use a mono-spaced font to view the art correctly (proportionally spaced fonts distort the layouts).
Use scanf to retrieve individual characters from the standard input and process them as you read them in (no need to store them in arrays).
Since the ASCII value of the character '0' is 48, you can convert a digit char to an int value (i.e. '1' => 1, '2' => 2 etc) using:

```
intValue = ((int) inputChar) - 48;
```

(though remember, we don't like to see unexplained magic numbers in your code, so make sure you use a constant or add a comment).

Part 2

Title: Sorted count of the unique words in a piece of text

Description

Write a C program called assignment2_2.c, that reads in some input text (e.g. a poem), which features a series of lines (each of no more than 100 characters in length) comprising zero or more words (each word is of at most 30 characters in length) in ASCII (which may include some punctuation characters that must be removed). This input text will contain an unknown number of unique words (each of which may appear several times in the input text). Your C program should process that text into a series of individual words (i.e. strings separated from one another by space, comma or full-stop characters), filter any non-alpha characters from the words, and store them in an appropriate data structure in your program.

The output should be an alphabetically sorted series of words (which will be composed of lowercase characters). Each unique word (along with its associated count) should be output. The sorting of the list of words should be achieved by a function of your own (do not use a library or third-party function to sort the words).

Think carefully about the memory efficiency of your code in order to achieve the highest marks for your solution. You should malloc the storage for the individual strings as you need them, storing them into your data structure. Depending on the data structure you use, you may also need to malloc space for it as it grows.

Input

A number of lines of text (each of 100 characters at most, featuring 0 or more words of at most 20 characters each).

Output

An alphabetically sorted list of search words and their frequency counts. Each line of the output should be in the following format: a unique word from the input text, followed by a space, a "=>" string another space, and then an integer count of how many times that word was found in the text.

Sample Input

```
The Jabberwock
Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
All mimsy were the borogoves,
And the mome raths outgrabe.
Beware the Jabberwock, my son!
The jaws that bite, the claws that catch!
Beware the Jubjub bird, and shun
The frumious Bandersnatch!
He took his vorpal sword in hand;
Long time the manxome foe he sought
So rested he by the Tumtum tree
And stood awhile in thought.
And, as in uffish thought he stood,
The Jabberwock, with eyes of flame,
Came whiffling through the tulgey wood,
And burbled as it came!
One, two! One, two! And through and through
The vorpal blade went snicker-snack!
He left it dead, and with its head
He went galumphing back.
And hast thou slain the Jabberwock?
Come to my arms, my beamish boy!
O frabjous day! Callooh! Callay!
He chortled in his joy.
Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
All mimsy were the borogoves,
And the mome raths outgrabe.
```

Sample Output

```
all => 2
and => 14
arms => 1
as => 2
awhile => 1
back => 1
bandersnatch => 1
beamish => 1
beware => 2
bird => 1
bite => 1
blade => 1
borogoves => 2
boy => 1
brillig => 2
burbled => 1
by => 1
callay => 1
callooh => 1
came => 2
catch => 1
chortled => 1
claws => 1
come => 1
day => 1
dead => 1
did => 2
eyes => 1
flame => 1
foe => 1
frabjous => 1
frumious => 1
galumphing => 1
gimble => 2
gyre => 2
hand => 1
hast => 1
he => 7
head => 1
his => 2
in => 6
it => 2
its => 1
jabberwock => 4
jaws => 1
joy => 1
jubjub => 1
left => 1
long => 1
manxome => 1
mimsy => 2
mome => 2
my => 3
o => 1
of => 1
one => 2
outgrabe => 2
raths => 2
rested => 1
shun => 1
slain => 1
slithy => 2
snickersnack => 1
so => 1
son => 1
sought => 1
stood => 2
sword => 1
that => 2
the => 20
thou => 1
thought => 2
through => 3
time => 1
to => 1
took => 1
toves => 2
tree => 1
tulgey => 1
```

```
tumtum => 1
twas => 2
two => 2
uffish => 1
vorpal => 2
wabe => 2
went => 2
were => 2
whiffling => 1
with => 2
wood => 1
```

HINTS

It is easier to compare two strings if they are converted to the same case (uppercase or lowercase) first.

Your program needs to process text until an EOF is encountered (indicating the end of the text). When testing your program yourself (not via Code Grade), it is difficult to generate an EOF on some platforms. It is simpler to prepare your test input data in a text file, and then pipe that data into your program using "<" e.g. if your program has been compiled into the a.out file then:

```
a.out < sourcetext.txt
```

will send the contents of sourcetext.txt into your program, and when all the contents of that file have been processed, your program will receive an EOF on the input stream.

You might find it easier to develop your program in stages. You could start by writing a program to read the input text and write out each word as it retrieves it from the input stream. You might find the string library function `strtok()` useful for this. When you can successfully do this, the next stage is to store each word in a data structure in your program.

e.g. each word could be maintained in a C struct that holds both a string and an integer count of the number of occurrences of the string.

Marks will be awarded for good and efficient use of structures and pointers and for not using unnecessary storage space.