separator.c

Overview:

Write a C program called separator.c that:

— Lets the user ***safely*** enter a string.

— Then parses that string into a linked list of smaller strings.

— The smaller strings come from the strings between the commas in the string the user entered.

For example, if the user types:

```
12345,1234,123,12,1,,
```

Then the program should separate it into 7 strings:

```
"12345"
"1234"
"123"
"12"
"1"
“”
“”
```

And then:

— Save the sequence of strings as a ***linked list.***

— Print the ***linked list***.

— `free()`'s the ***linked list.***

Sample output:

```
$ ./separator
Please enter a line of text: Mary,had,a,little,lamb
     "Mary"
     "had"
     "a"
     "little"
     “lamb"


$ ./separator
Please enter a line of text:  ,,   ,,
     “”
     “”
     “ “
     “”
     “”
```

```
$ ./separator
Please enter a line of text: 12345,1234,123,12,1, ,
     "12345"
     "1234"
     "123"
     "12"
     "1"
     ""
     ""
```

Coding:

You **must** put your strings in the following **C (not C++)** list node:

```
struct         Word
{
  char*        textPtr_;
  struct Word* nextPtr_;
};
```

NOTE: you may only use the following C string functions (of course you will not need them all):

`printf(), fprintf(), malloc(), calloc(), realloc(), free(), fgets(), snprintf(), strncpy(), strncat(), strncmp(), strdup(), strlen() and strchr()`

HINT: I recommend writing 4 functions:

— `struct Word* obtainCommaSeparatedList (const char* string)`

— `void printCommaSeparatedList (const struct Word* list)`

— `void freeCommaSeparatedList (struct Word* list)`

— `int main()` (of course!)

MORE HINTS:

— To get rid of the annoying `\n` char that `fgets()` adds to the end of entered strings, do something like:

```
char*    cPtr  =  strchr(line, '\n');

if  (cPtr != NULL)
  *cPtr = '\0';
```

— Have a `char*` variable (e.g. charRun) that starts at the beginning of the next string (e.g. `start`). Advance `charRun` to the next comma or null character (noting it may already be at one). Then use pointer arithmetic to determine how many chars separate the two (`charRun - string`)

— To make a new heap-allocated `struct Word` node, say:

```
struct Word*   toReturn  = (struct Word*)malloc(sizeof(struct Word));
```