# String Processing and Memory in C

This is an individual assignment, not a group effort. You can use any of the C library functions. The point of this project is to help you get better with using strings and memory allocation in C. The majority of the work is expected to be your own, see the Department’s plagiarism case studies if in doubt.

## Description

The “Lorem Ipsum” typesetter’s text is a common source of text for different applications that just need a place holder. In fact, an approximately 3KB string of text is available from this git repo:

<https://github.com/ccpalettes/sublime-lorem-text.git>

The contents of this file is a single long line, in the form (this is just the first few hundred characters).

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim. Fusce est.

Your job is to write a program that converts this to sentences. A *sentence* is defined here to be any string of characters that ends with (and includes) the “period” (ASCII 0x2E).

Your task is to read a single line of input from *standard input*, convert it to strings, and display the number of strings found, and then each string. Since we don’t know ahead of time how many sentences you will have, you should count the number of periods in the input first. Then use malloc() to create an array of string pointers, and as you find strings, you should use malloc() to create a place to copy each string, then copy the string that you’ve found, and go onto the next.

After you are done, you will need to free all of the memory you’ve allocated (in the right order / right way). You can / should use “valgrind” for this (I know the autograder will).

One more tip is that **good style** says that you should keep related data items together (usually in a struct), so use a struct to hold the *number of sentences* and the *array of strings*.

## Sample Input & Output

$make clean ; make all

gcc -o sentences sentences.c  
$ echo "I'm sorry Dave. I cannot do that." | ./sentences  
num: 2  
0 : (I'm sorry Dave.)  
1 : (I cannot do that.)

The maximum length of input will be 4096 (4KB).

## Deliverables

You will upload your *source code* to Gradescope. An autograder will run several tests on it. After the due date has passed then I will review and grade your source code. I am looking specifically for the following three areas:

* Proper use of malloc() / free() (or equivalent)
* Proper code style – proper use of built-in and user-defined data types, function names, comments, readability
* Proper functional decomposition – using functions to break down complex tasks into smaller, easier to code, easier to understand tasks.

You shall include only a Makefile that creates an executable called “sentences” when run as “make all”, and your needed source files.

The autograder code will take whatever files you upload, strip out executables (and report an error). The autograder will invoke your “make all”, and then run your “sentences” executable.