CSCI 352 Unix Software Development
Summer 2015
C Programming Warm-up Task 2

The purpose of this warm-up task is to get you started in C programming on a fairly simple task, but one that builds a component that will be needed for the first assignment. You may, but are not required to, submit your program for evaluation and critique. You are encouraged to submit your program multiple times so that you may improve it from feedback.

Function gettokens()

Your task is to write a function called gettokens(), declared as:

```
char **gettokens(char *line);
```

Given the string line as a parameter, the function is to return an NULL-terminated array of substrings from line.

The substrings are separated in line by spaces, but double quotes may be used to include a space within a substring. You may assume that unclosed double quotes extend the substring to the end of the line. A double quote within a token extends that token to the next double quote or the end of the line.

For example:

Line	Substrings
this is an example	this is an example
this is “an example”	this is an example
this is ”an example	this is an example
this i”s an” example	this isan example

Supplied Files

The following files are provided:

tokenizer.h	header file containing the declaration of gettokens()
tokentest.c	a test program which uses gettokens()
Makefile	makefile for you to use in building the executable

Coding Standards

1. Use meaningful names that give the reader a clue as to the purpose of the thing being named.
2. Avoid the repeated use of numeric constants. For any numeric constants used in your program, use #define preprocessor directives to define a macro name and then use that name wherever the value is needed.
3. Use comments at the start of the program to identify the purpose of the program, the author and the date written.
4. Use comments at the start of each function to describe the purpose of the function, the purpose of each parameter to the function, and the return value from the function.
5. Use comments at the start of each section of the program to explain what that part of the program does.
6. Use consistent indentation.