In this assignment, you will write a program that, when given two strings, finds their longest common substring (LCS); that is, the longest substring contained within both strings. For instance, if \( s = \) “periscope” and \( t = \) “experience”, the LCS of \( s \) and \( t \) is “peri”. The LCS could be of length 0 (if the strings have no characters in common), or could be equal to one (or both) of the two strings. Two strings might also have multiple LCSs (e.g., “human being” and “being human” have the LCSs “human” and “being”); for this assignment, either result can be reported.

Let \( s = s_0 s_1 \cdots s_{m-1} \) and \( t = t_0 t_1 \cdots t_{n-1} \) be two strings of length \( m \) and \( n \), respectively. To find the longest common substring, one can compute an array \( L \) where for every \( i \in \{0, m-1\} \) and every \( j \in \{0, n-1\} \), the length \( L_{ij} \) of the longest common substring ending with \( s_i \) and with \( t_j \). For instance, for \( s = \) “periscope” and \( t = \) “experience”, \( L_{3,5} = 4 \), indicating that \( s_{0..3} = t_{2..5} = \) “peri”. To compute the \( L \) array, one can iterate through both strings, computing \( L_{ij} \) by adding 1 to \( L_{i-1,j-1} \) (or setting it to 1 if \( i \) or \( j \) is 0) if the strings continue to match. Using the example strings, \( L \) is computed to be:

\[
\begin{array}{cccccccccc}
& e & x & p & e & r & i & e & n & c \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 1 \\
2 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 1 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 1 \\
\end{array}
\]

Once \( L \) is computed, the LCS is easy to find—just find the largest element of \( L \) (call it \( \ell \)) at position \( (i, j) \) in the array; the LCS is \( s_{i-\ell+1} \cdots s_i \) (or alternatively \( t_{j-\ell+1} \cdots t_j \)). For more efficiency, one can keep track of the longest substring seen so far while computing \( L \).

1. Write a function that takes two input strings and an output buffer of type char*, finds (one of) the longest common substring of the two input strings, and copies it into the output buffer. Be sure to null terminate the output buffer so it is a legal C string.

2. Write a function \texttt{main} that takes two arguments (plus the name of the program) on the command-line, invokes the function of part 1 on these arguments and an output buffer, and then prints the result to stdout. You may assume the LCS is at most 100 characters long. You may use the following template.

```c
int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf(stderr, "usage: %s <string1> <string2>
", argv[0]);
        exit(1);
    }
/* insert your code here */
    return 0;
}
```
Your program should be in one .c file. The function of part 1 should be after main in the file, so remember to correctly declare a prototype for the function before main. #include any header files for library functions you need to call. You may develop the code in Visual Studio or any other environment you choose, but please be sure your code compiles and runs on omega with gcc.

You will be graded on the program correctness as well as the use of good coding style. In particular:

- Use consistent indentation.
- No more than one statement per line.
- Use whitespace to separate declarations and statements.
- Use meaningful identifiers. Constants should be upper case and should be defines with either a #define or with a const declaration.
- Prototype all functions other than main.
- Declare important constants rather than using the literal value.

Help for all assignments is available from the lab instructors or from Prof. Nystrom during his office hours.

**Turn in.** Email your program on or before the due date (Nov 16, 5pm) to nystrom@uta.edu with the subject “CSE 1310 turnin N1”.