csce350 — Data Structures and Algorithms
Fall 2019 — Project 4

Assigned: November 20
Due: December 5, 11:54pm

For this assignment only, late submissions will be accepted without penalty until 11:54pm on December 8. No submissions after December 8 will be accepted.

The purpose of this assignment is to give you some practice designing and implementing dynamic programming algorithms.

The Problem Hannah and her friend Otto like palindromes, which you will probably recall are strings that read the same both forward and backward. In fact, they like palindromes so much that, when they see a word that is not a palindrome, they wonder about how to *cut* that word into smaller strings that are palindromes. That's why our friends want a program that will read words and *compute the smallest number of cuts* needed to turn each word into a palindrome. Here are some examples.

Original String	Number of Cuts	Partition into Palindromes
abb	1	a|bb
abbc	2	a|bb|c
abba	0	abba
abbab	1	abba|b
abcde	4	a|b|c|d|e

If the original word is a palindrome already, no cuts at all are needed. Perhaps less obviously, notice that it's always possible to cut any word into palindromes — this works because single characters count as palindromes.

Your Task You should write a C++ program that does these things:

1. Read a string, consisting of a sequence lowercase letters, from standard input.
2. Compute the smallest number of cuts needed to separate the string into palindromes. Print a line containing this number, followed by a space and then the string itself with pipe characters (|) inserted at the locations of those cuts. For full credit, your program should do this in $O(n^2)$ time for input strings of length n.
3. Return to Step 1 above to read and cut another string. Terminate when standard input reaches end-of-file.

Sample Input 1	***Sample Output 1***
`abb`	`1 a|bb`
`abbc`	`2 a|bb|c`
`abba`	`0 abba`
`abbab`	`1 abba|b`
`abcde`	`4 a|b|c|d|e`

Sample Input 2	*Sample Output 2*
donatello	7 d|o|n|a|t|e|ll|o
finn	2 f|i|nn
papa	1 p|apa
twilight	5 t|w|ili|g|h|t

Sample Input 3

```
gxrrxgxueeuxuudmmdpptt
gxrrxgxueeuxuudmmdpptttttppdmmduuxueeuxgxrrxg
ddxqqxzijjizxxwuiiuwvxmmxv
ddxqqxzijjizxxwuiiuwvxmmxvvxmmxvwuiiuwxxzijjizxqqxdd
ivllvixxmwttwmxjjxksbbsk
ivllvixxmwttwmxjjxksbbskksbbskxjjxmwttwmxxivllvi
```

Sample Output 3

```
5 gxrrxg|xueeux|uu|dmmd|pp|tt
0 gxrrxgxueeuxuudmmdpptttttppdmmduuxueeuxgxrrxg
5 dd|xqqx|zijjiz|xx|wuiiuw|vxmmxv
0 ddxqqxzijjizxxwuiiuwvxmmxvvxmmxvwuiiuwxxzijjizxqqxdd
4 ivllvi|xx|mwttwm|xjjx|ksbbsk
0 ivllvixxmwttwmxjjxksbbskksbbskxjjxmwttwmxxivllvi
```

Hints A few possibly helpful comments:

- You may find it helpful to start by writing code to determine, for each possible substring, whether that substring is a palindrome or not, and store those results in a two-dimensional array of boolean values. The brute force way of doing this would take $\Theta(n^3)$ time, but with some care you can get it down to $\Theta(n^2)$. (Perhaps you might start with substrings of length 1, then length 2, and so on?) The results of this process can be very useful for solving the main problem.
- You may find it helpful to focus on counting the number of cuts needed, rather than inserting the cuts into the string. If you can compute the number correctly, then getting the cuts themselves is not likely to require huge changes.
- For calibration purposes, my solution has 65 lines of code, including some comments. This can be a very short program, but you'll need to think, design, and debug very carefully.
- (Added 2019-11-22) A few students have asked about the $O(n^2)$ time requirement. You should, by this point in the class, be able to analyze your algorithm to tell whether it meets this constraint. But just in case you have some doubts, a test input file called `project4-5.in` has been published on the course website. One particular $\Theta(n^2)$ time implementation, running on my laptop, solves this instance in about 0.05 seconds; a $\Theta(n^3)$ time implementation takes about 60 seconds to compute the same answer on the same computer. If your program can solve `project4-5.in` quickly (and correctly), then your algorithm is probably fast enough to meet the time requirement.

What to Submit You should submit, using the department's dropbox website, a single C++ source file named containing all of the code for your program. I will compile this program using this command line:

```
g++ -Wall -std=c++11 yourfile.cpp
```