COMP9101 Ass03
Name: Zanning Wang
zID: z5224151

1. Boolean operators NAND and NOR are defined as follows

NAND	*true*	*false*
true	*false*	*true*
false	*true*	*true*

NOR	*true*	*false*
true	*false*	*false*
false	*false*	*true*

You are given a boolean expression consisting of a string of the symbols *true*, *false*, separated by operators AND, OR, NAND and NOR but without any parentheses. Count the number of ways one can put parentheses in the expression such that it will evaluate to *true*. (20 pts)

1.

In order to solve this problem, we may use dynamic programming to solve this problem.
Suppose we have T(i, j), which means we have the number of ways to parenthesize the symbols from i to j to get TRUE result. The same for F(i, j), we have the number of ways to parenthesize the symbols from i to j to get FALSE result.

For the base case, we use cur_sym to show the result of current symbol, therefore,
When cur_sym[i] === True, obviously T[i][i] = 1, F[i][i] = 0;
When cur_sym[i] === False, obviously F[i][i] = 0, F[i][i] = 1;

For T(i, j), the problem can be divide into the sub-problem T(i, k) and T(k+1, j), k∈[i, j];
For F(i, j), the problem can be divide into the sub-problem F(i, k) and F(k+1, j), k∈[i, j];

Suppose the symbol is "AND":
In order to get the TRUE result, the result of each sub-problem should be TRUE
Therefore the T(i, j) = T(i, k) * T(k+1, j), which k∈[i, j];

Suppose the symbol is "OR":
In order to get TRUE result, at least one of sub-problem should be TRUE is enough
Both TRUE: T(i, j) => T(i, k) * T(k+1, j), which k∈[i, j];
One TRUE: T(i, j) => T(i, k) * F(k+1, j) + F(i, k) * T(k+1, j), which k∈[i, j];
Therefore the T(i, j) = T(i, k) * T(k+1, j) + T(i, k) * F(k+1, j) + F(i, k) * T(k+1, j), which k∈[i, j];

Suppose the symbol is "NAND":
According to the definition, only TRUE NAND TRUE will get the False, all other situations will get the TRUE, therefore the result show below:
T(i, j) = F(i, k) * T(k+1, j) + T(i, k) * F(k+1, j) + F(i, k) * F (k+1, j);

Suppose the symbol is "NOR":
According to the definition, only TRUE NAND TRUE will get the TRUE;

Therefore T(i, j) = F(i, k) * F(k+1, j);

Therefore the recursion show bolow:

$$T(i,j) = \sum_{k=i}^{j-1} \begin{cases} \mathrm{T(i,k)} * \mathrm{T(k+1,j)} & cur_{sym[k]} = AND \\ \mathrm{T(i,k)} * \mathrm{T(k+1,j)} + \mathrm{T(i,k)} * \mathrm{F(k+1,j)} + \mathrm{F(i,k)} * \mathrm{T(k+1,j)} & cur_{sym[k]} = OR \\ \mathrm{F(i,k)} * \mathrm{T(k+1,j)} + \mathrm{T(i,k)} * \mathrm{F(k+1,j)} + \mathrm{F(i,k)} * \mathrm{F\,(k+1,j)} & cur_{sym[k]} = NAND \\ \mathrm{F(i,k)} * \mathrm{F(k+1,j)} & cur_{sym[k]} NOR \end{cases}$$

The same for F(i, j), we can get the recursion show bolow:

$$F(i,j) = \sum_{k=i}^{j-1} \begin{cases} \mathrm{T(i,k)} * \mathrm{T(k+1,j)} + \mathrm{T(i,k)} * \mathrm{F(k+1,j)} + \mathrm{F(i,k)} * \mathrm{F(k+1,j)} & cur_{sym[k]} = AND \\ \mathrm{T(i,k)} * \mathrm{T(k+1,j)} & cur_{sym[k]} = OR \\ \mathrm{T(i,k)} * \mathrm{F(k+1,j)} + \mathrm{F(i,k)} * \mathrm{T(k+1,j)} + \mathrm{T(i,k)} * \mathrm{T\,(k+1,j)} & cur_{sym[k]} = NOR \\ \mathrm{T(i,k)} * \mathrm{T(k+1,j)} & cur_{sym[k]} NAND \end{cases}$$

Therefore we can count the number of TRUE result by T(i, j) show above.