Jin-Ao Olson Zhang
Z52ii4i4
Question 1.

Suppose True(i,j) which represents the number of ways to place parentheses between i and j that gives True and False(i,j) represents the number of ways to place parentheses between i and j that gives False
Base case：
True(i,i) = 1 if i = T
True(i,i) = 0 if i = F
False(i,i) = 0 if i = T
False(i,i) = 1 if i = F

So for every expression E with size e, it can be expressed as

Expression1(i,k) &/||/NAND/NOR expression2 (k+i,e) , i<=k<=e

&:
If expression1(i,k) & expression2 (k+i,e) == True
Expression1 == expression2 = True
Which is True(i,k) & True(k+i,e) ways
||:
If expression1(i,k) & expression2 (k+i,e) == True
It could be
True(i,k) & True(k+i,e)
False(i,k) & True(k+i,e)
True(i,k) & False(k+i,e)
NAND:
opposite to &
ALL - True(i,k) & True(k+i,e) == False(i,k)*False(k+i,e)+False(i,k) & True(k+i,e)+True(i,k) & False(k+i,e)
NOR:
Opposite to ||
ALL - True(i,k)*True(k+i,e)+False(i,k) & True(k+i,e)+True(i,k) & False(k+i,e)
= False(i,k)*False(k+i,e)

If expression1(i,k) & expression2 (k+i,e) == False
For all above just calculate
True(i,e) - expression1(i,k) & expression2 (k+i,e) == True

So recursively

True(i,j) = $\sum_{k=i}^{j-1}$

If operator at K = &
    True(i,k) & True(k+i,j)
If operator at K = ||
    True(i,k)*True(k+i,j) + False(i,k) & True(k+i,j) + True(i,k) & False(k+i,j)
If operator at K = NAND
    False(i,k)*False(k+i,j) + False(i,k) & True(k+i,j) + True(i,k) & False(k+i,j)
If operator at K = NOR
    False(i,k)*False(k+i,j)

False(i,j) = $\sum_{k=i}^{j-1}$

If operator at K = &
    False(i,k)*False(k+i,j) + False(i,k) & True(k+i,j) + True(i,k) & False(k+i,j)
If operator at K = ||
    False(i,k)*False(k+i,j)
If operator at K = NAND
    True(i,k) & True(k+i,j)
If operator at K = NOR
    True(i,k)*True(k+i,j) + False(i,k) & True(k+i,j) + True(i,k) & False(k+i,j)

And total way between i and j will be T(i,j)
It takes O(n)