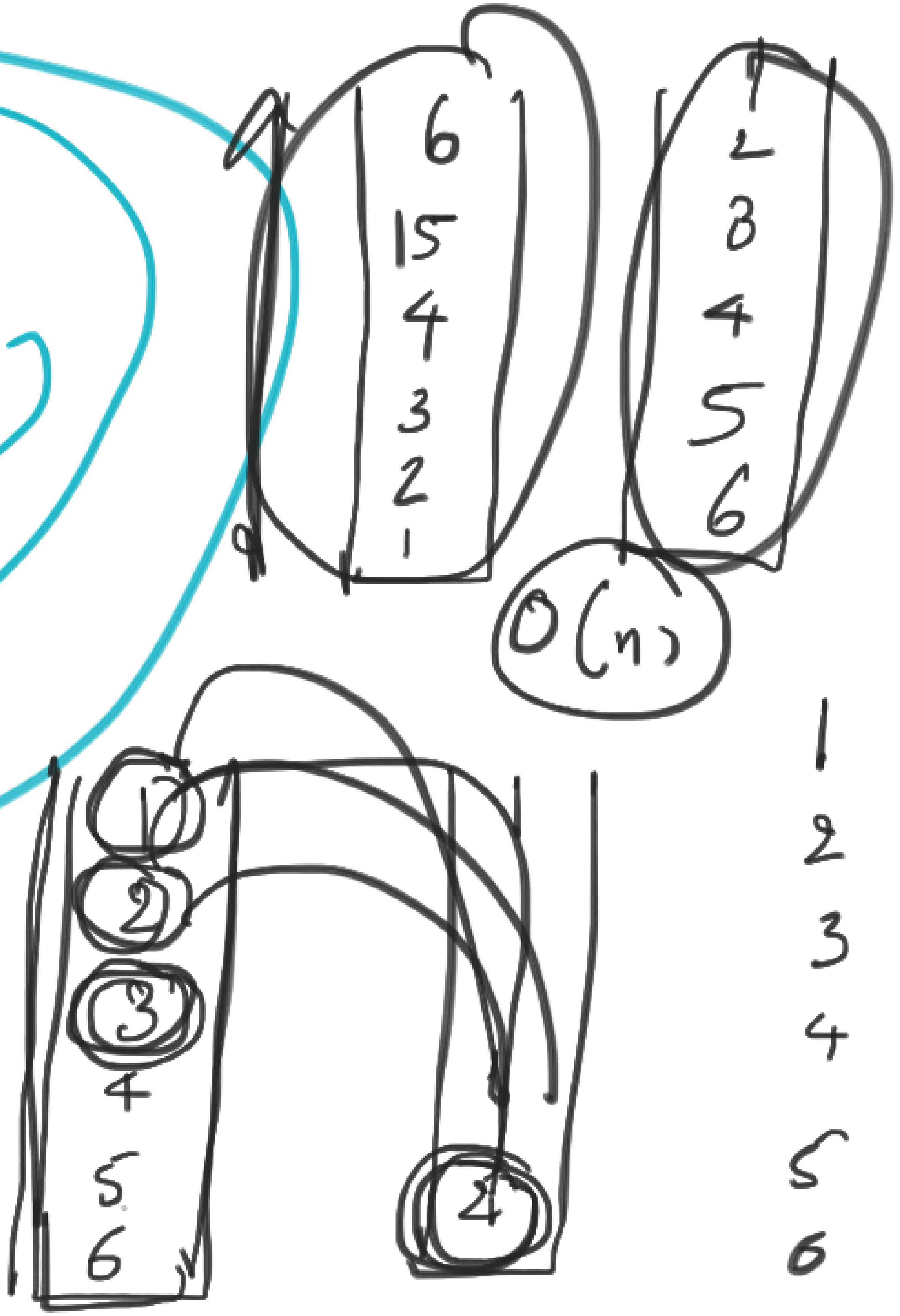
```
while (count>0)
{
    temp.push(s.top())
    s.pop()
    count = count-1
}
```

worst case scenario

$1+2+3+ --- . n$ $O(n^2)$



Sort stack using recursion. ?

exp []

n

n % 2 != 0

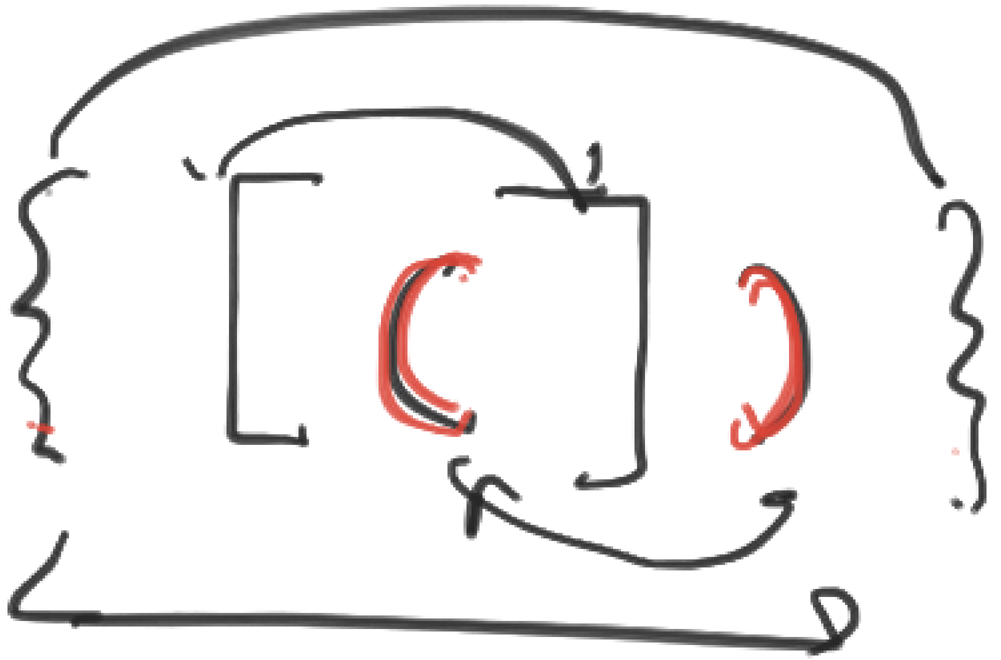
Odd

Valid parentheses

expression

balanced expression

({ [ ] })



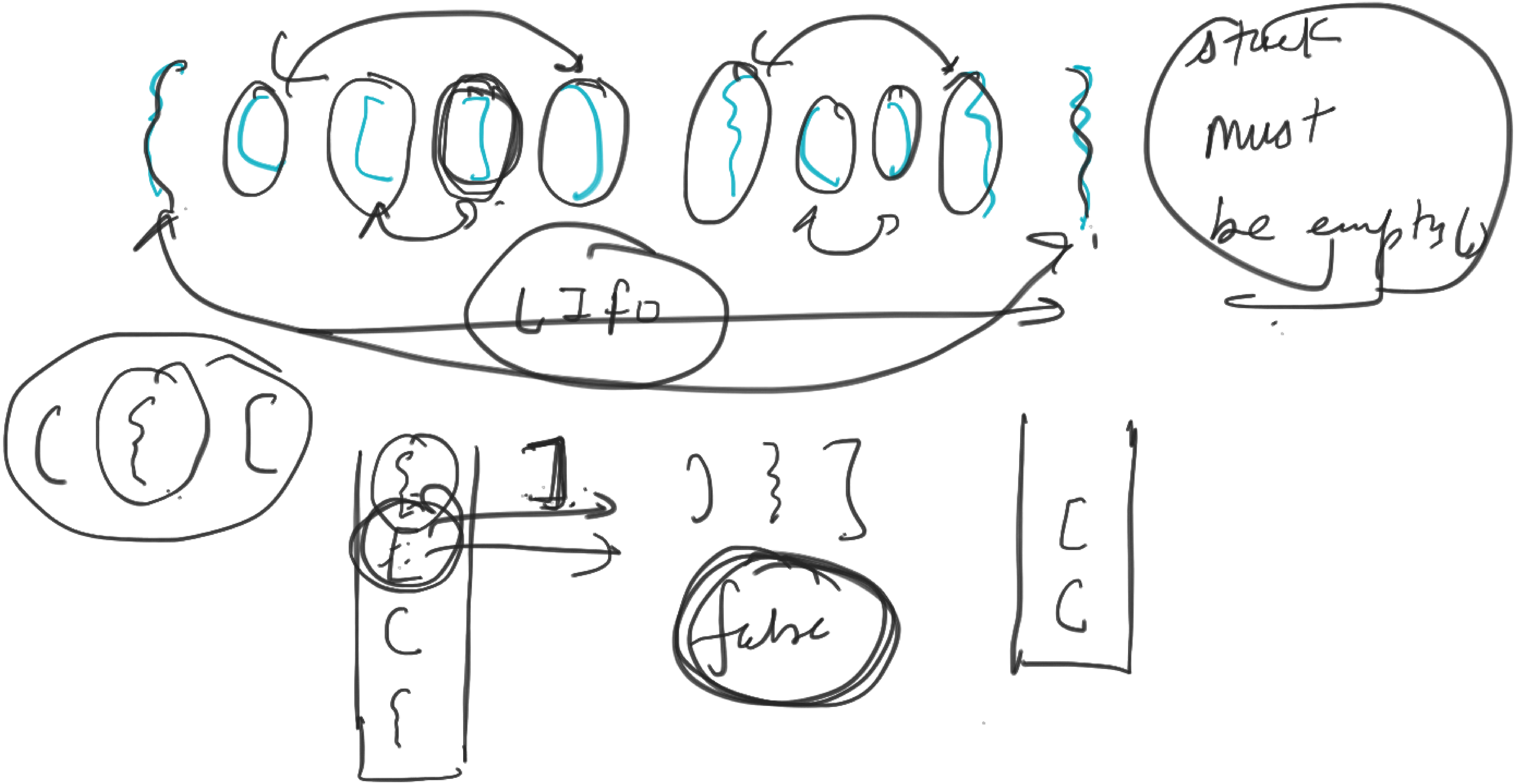
✗

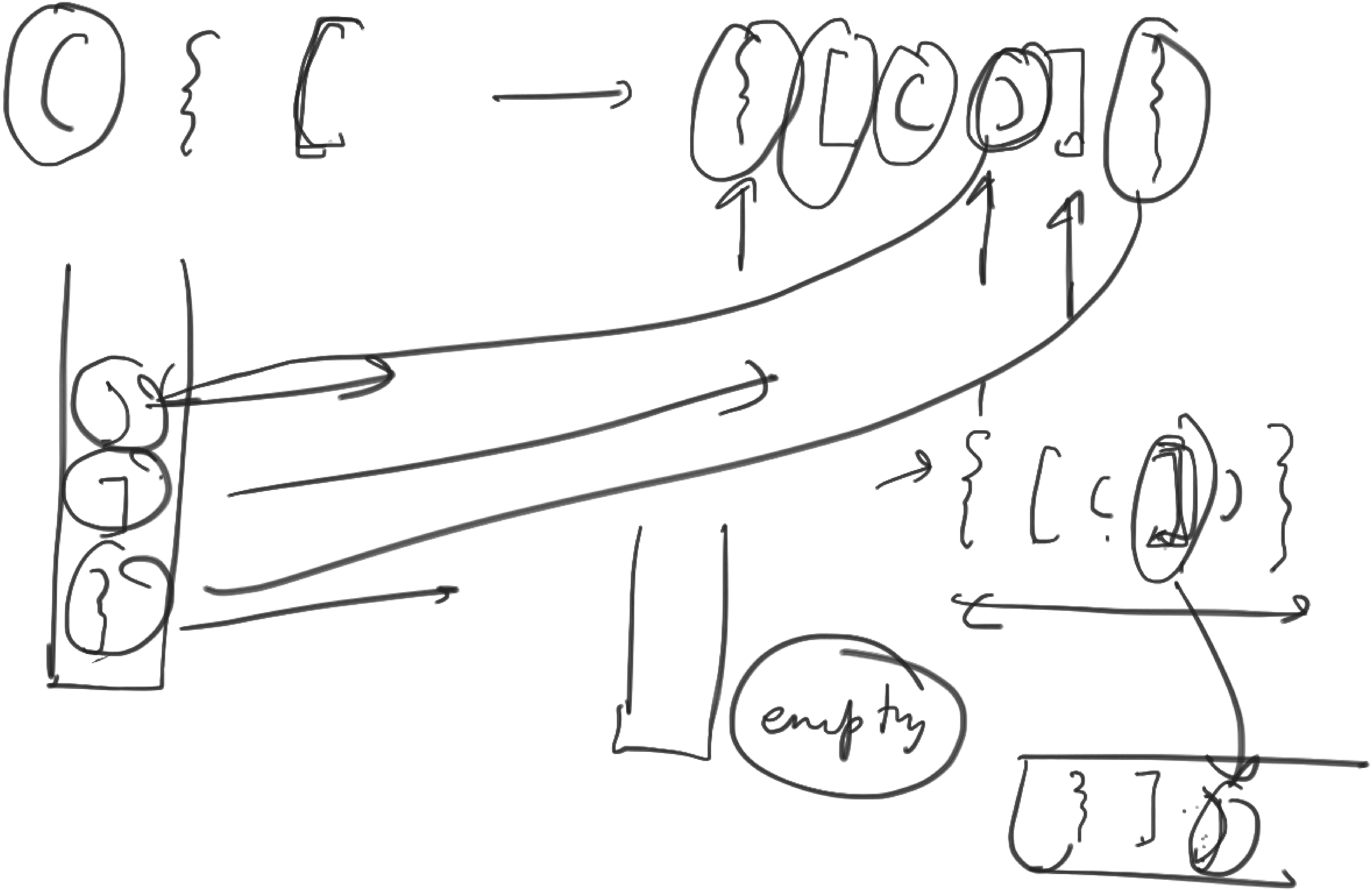
⇒ { [( )] }

order

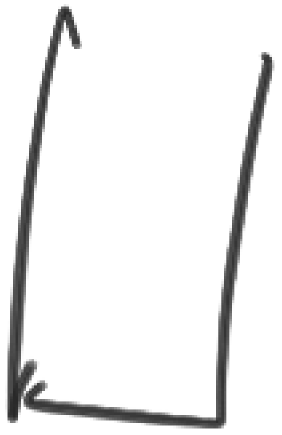
( )

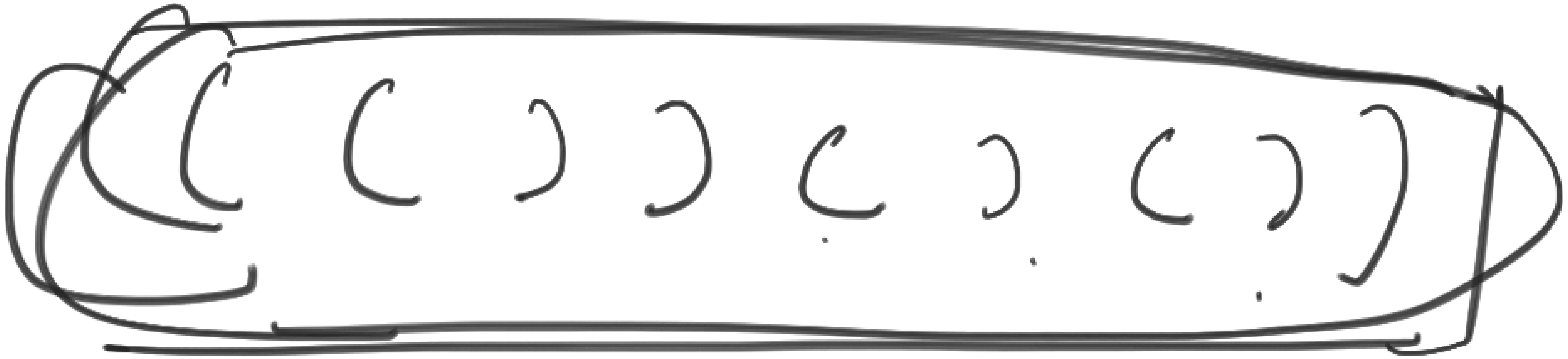
LIFO


{ ([ ]) { ( ) } }
stack must be empty()
LIfo
({ [
{ [({
]) }]
false
[(


({ [
{ [( )] }
)] }
{ [( ]) }
empty
}])

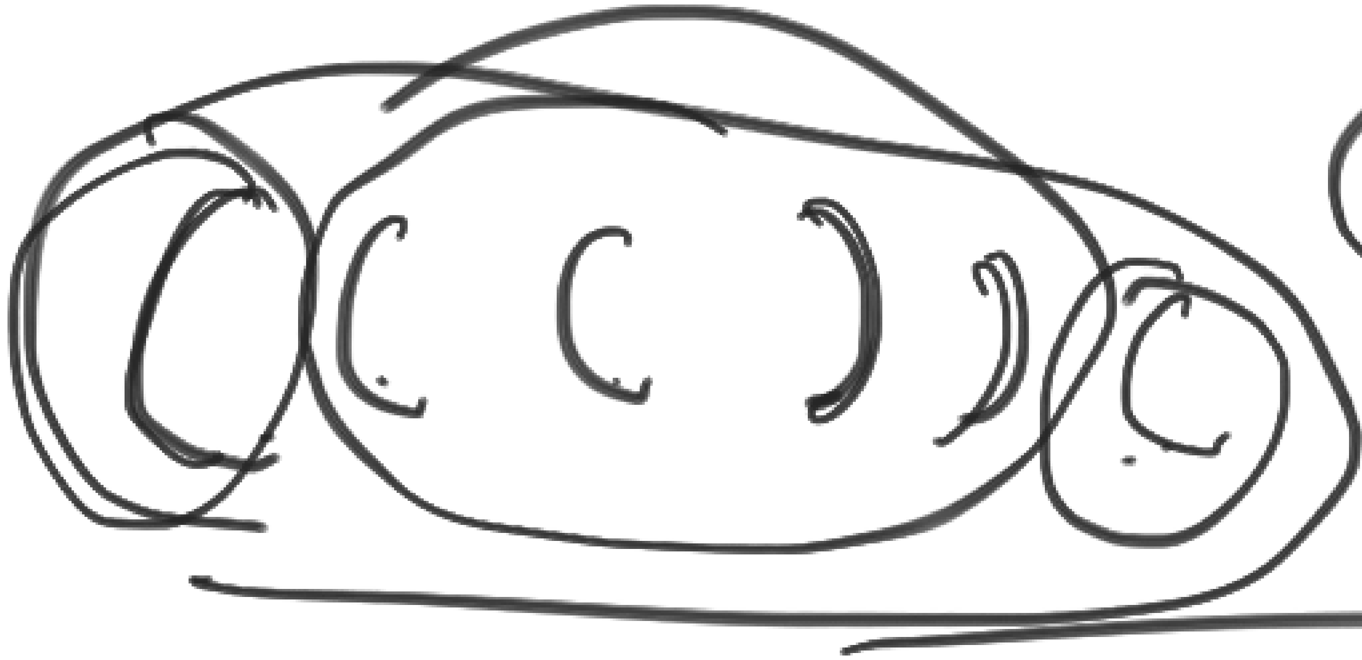
```
for (i = 0 to exp.size()-1)
{
    if ( exp[i] == ( )
        s.push( ))
    ______ == {
        s.push( })
    ______ == [
        s.push( ])
    else if
    ( !s.empty()
    && s.top() == exp[i]
        s.pop()
    else
        return false)
}
return (s.entty())
```





without using stack

1

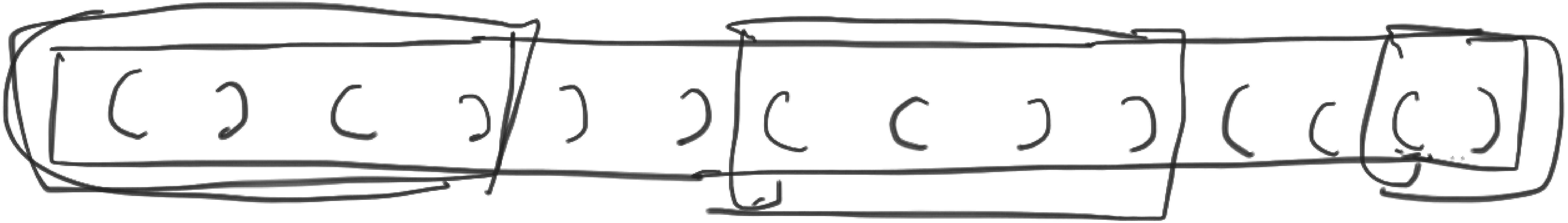


unbalanced

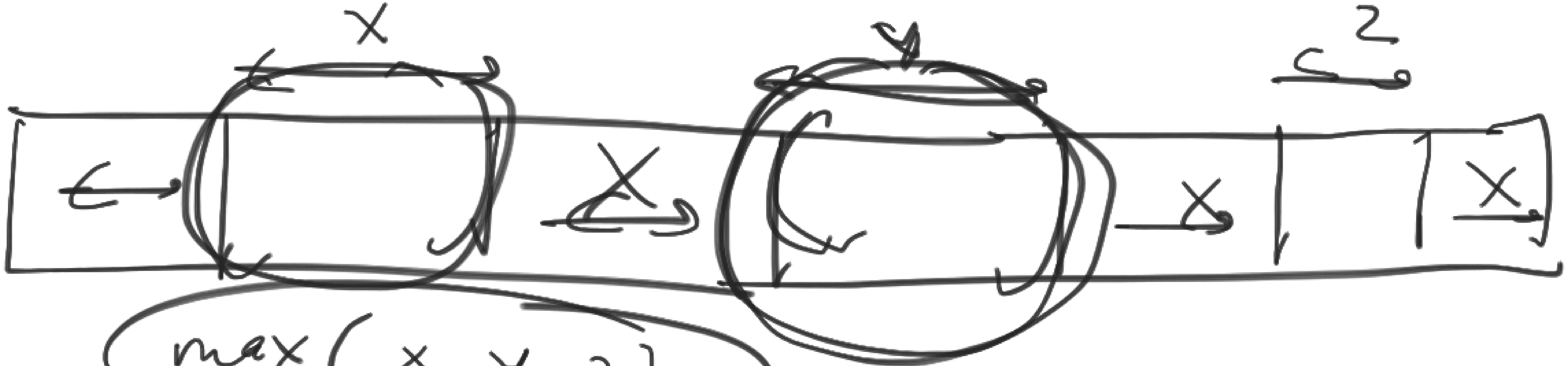
min no. of removal

balanced

min no. of additions



✓ longest valid parenthesis.



max(x, y, 2)

Generate all balanced Parenthesis.

of size n $\Rightarrow$ n $=$ 3

n n

( )

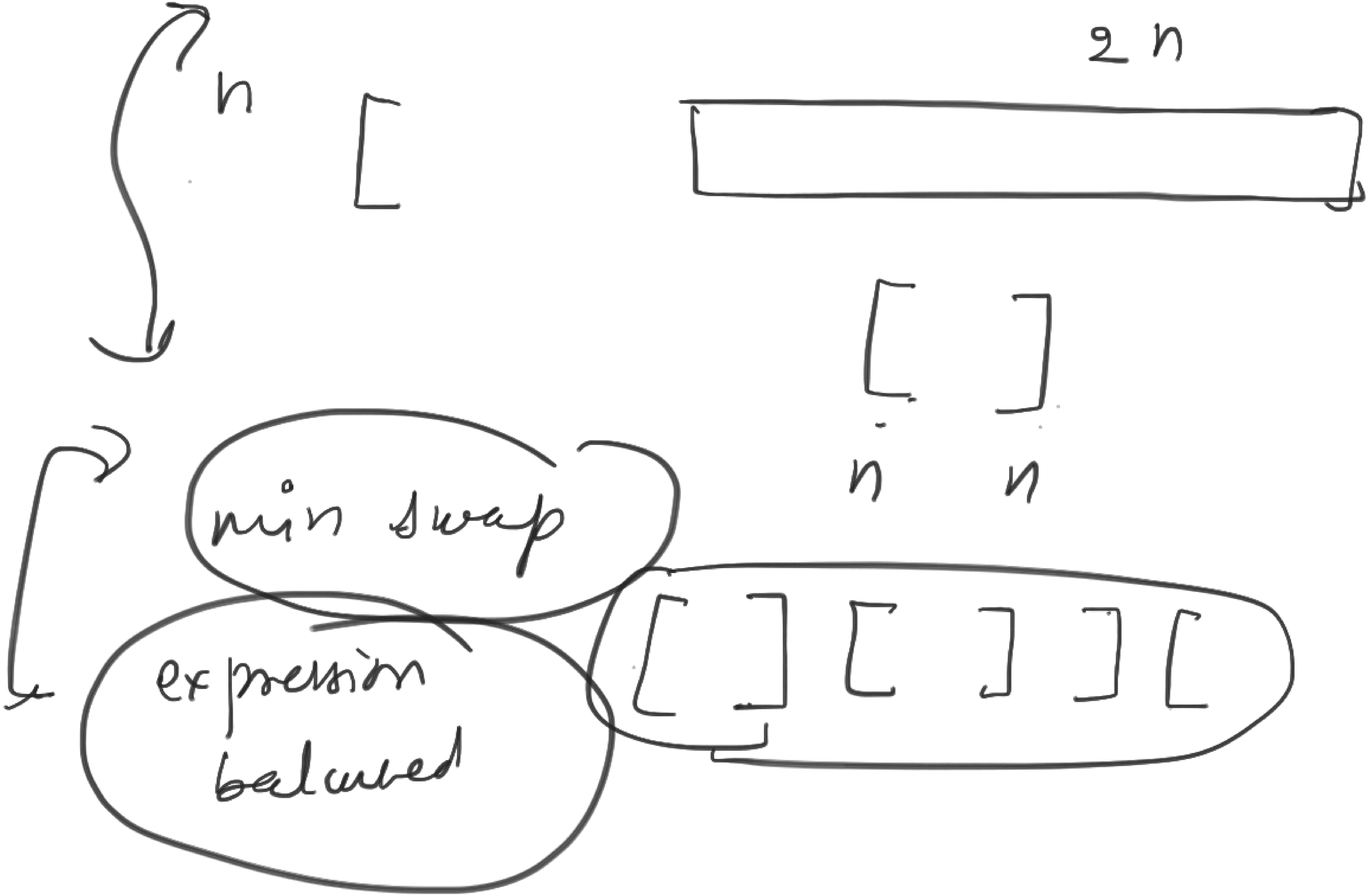
(())

()()

((()))

()()() ~~(()~~

(())() ()(())


n
[
2 n
[ ]
n n
min swap
expression balanced
[ ] [ ]] [