# HW1 - Regular Expression Parsing

Here is a set of strings with balanced parentheses.

In [28]:
```
yes1 = "x (a b c) y"
yes2 = "((((x))))"
yes3 = "a (b (c d) ((e)) f) (g)"
```

Here is a set of strings with unbalanced parentheses.

In [29]:
```
no1 = "(a b c"
no2 = "((())))"
no3 = "a (b (c d) (e)) f) (g)"
```

Write a function `check_balanced` that uses regular expressions to check whether the parentheses are balanced. Note that you can't do this with a single regular expression, you need to write a little loop around it. Your code structure might differ a little from the function below, but it shouldn't be much longer.

In [2]:
```
import re
def check_balanced(s):
    number_of_subs_made = 1  # temp number
    pattern1 = "\([^\(\)]+\)"
    repl1 = 'x'  # just a filler
    while number_of_subs_made > 0:
        (new_string, number_of_subs_made) = re.subn(pattern1,repl1,s)
        s = new_string
    pattern2 = "[^\(\)]"  # selects all the non parentheses
    repl2 = ''  # empty string
    (new_string, number_of_subs_made) = re.subn(pattern2,repl2,s)
    return len(new_string) == 0
```

Now show that it works.

In [31]:
```
print check_balanced(yes1)
print check_balanced(yes2)
print check_balanced(yes3)
```

```
True
True
True
```

In [32]:
```
print check_balanced(no1)
print check_balanced(no2)
print check_balanced(no3)
```
```
False
False
False
```

In [3]:
```
print check_balanced(')()(')
```
```
False
```

In [ ]:
```
```