In [16]:
```
def generateParenthesis(n):
    return(genP(n))
def genP(n,l=0,r=0,cur="",result=[]):
        if(n==0):
            #result.append(cur)
            #return(result);
            return([''])
        #print([''])
        else:
            if (l == n & r == n):
                result.append(cur);
                return;

            if (l < n):
                newCurrent = cur + "(";
                genP(n, l + 1, r, newCurrent, result)

            if ((r < n) & (l > r)):
                newCurrent = cur + ")";
                genP(n, l, r + 1, newCurrent, result)
            return(result)

generateParenthesis(0)
```

Out[16]:
```
['']
```

In [17]:
```
generateParenthesis(1)
```

Out[17]:
```
['()']
```

In [18]:
```
generateParenthesis(2)
```

Out[18]:
```
['()', '(())', '()()']
```

In [19]:
```
generateParenthesis(3)
```

Out[19]:
```
['()', '(())', '()()', '((()))', '(()())', '(())()', '()(())', '()()()']
```

In [20]:
```
def isPalindrome(st):

    st = st.lower()
    st = ''.join(char for char in st if char.isalnum())
    f=True
    m= len(st)//2

    for i in range(m):
        if(st[i] != st[len(st)-i-1]):
            f = False;
            break;

    if(f):
        return(f)
```

```
    else:
        return(f)

isPalindrome("race a car")
```

Out[20]: False

In [21]:
```
isPalindrome("A man, a plan, a canal: Panama")
```

Out[21]: True

In [ ]:
```

```