In [86]:

```
import numpy as np
import math
import pandas as pd
```

In [87]:

```
class Node:
    def __init__(self, attribute):
        self.attribute = attribute
        self.children = []
        self.answer = ""

    def __str__(self):
        return self.attribute
```

In [88]:

```
def subtables(data, col):
    dict = {}
    items = np.unique(data[:, col])

    count = np.zeros((items.shape[0], 1), dtype=int)

    for x in range(items.shape[0]):
        #count[x] = sum(data[:,col] == items[x])
        for y in range(data.shape[0]):
            if data[y, col] == items[x]:
                count[x] +=1
            #pass

    for x in range(items.shape[0]):
        dict[items[x]] = np.empty((int(count[x]), data.shape[1]), dtype="S32")
        pos = 0
        for y in range(data.shape[0]):
            if data[y, col] == items[x]:
                dict[items[x]][pos] = data[y]
                pos+=1
        # remove column
        dict[items[x]] = np.delete(dict[items[x]], col, 1)

    return items, dict
```

In [89]:

```
def entropy(S):
    items = np.unique(S)

    if items.size == 1:
        return 0

    counts = np.zeros((items.shape[0],1))
    sums = 0

    for x in range(items.shape[0]):
        counts[x] = sum( S == items[x]) / (S.size)

    for count in counts:
        sums += -1 * count * math.log(count, 2)

    return sums
```

In [90]:

```
def gainRatio(data, col):
    items, dict = subtables(data, col)

    totalSize = data.shape[0]

    entropies = np.zeros((items.shape[0],1))

    for x in range(items.shape[0]):
        ratio = dict[items[x]].shape[0]/totalSize
        entropies[x] = ratio * entropy(dict[items[x]][:, -1])

    totalEntropy = entropy(data[:,-1])

    for x in range(entropies.shape[0]):
        totalEntropy -= entropies[x]

    return totalEntropy
```

In [91]:

```
def createNode(data, metadata):
    if(np.unique(data[:,-1])).shape[0] == 1:
        node = Node("")
        node.answer = np.unique(data[:, -1])[0]
        return node

    gains = np.zeros((data.shape[1] - 1, 1))

    for col in range(data.shape[1]-1):
        gains[col] = gainRatio(data, col)

    split = np.argmax(gains)
    node = Node(metadata[split])
    metadata = np.delete(metadata, split, 0)
    items, dict = subtables(data, split)

    for x in range(items.shape[0]):
        child = createNode(dict[items[x]], metadata)
        node.children.append((items[x], child))

    return node
```

In [92]:

```
def readData(filename):
    data = pd.read_csv(filename, header=None)
    metadata = np.array(data.iloc[0, :])
    traindata = np.array(data.iloc[1:, :])

    return (metadata, traindata)
```

In [93]:

```
def empty(size):
    s = ''
    for x in range(size):
        s += "    "
    return s
```

In [94]:

```
def printTree(node, level):
    if node.answer!= "":
        print(empty(level), node.answer)

    print(empty(level), node.attribute)

    for value, n in node.children:
        print(empty(level+1), value)
        printTree(n, level + 2)
```

In [95]:

```
metadata, traindata  = readData("Pgm 3 TennisDT.csv")

node = createNode(traindata, metadata)

printTree(node, 0)
```

```
Outlook
    Overcast
        b'Yes'

    Rain
        Wind
            b'Strong'
                b'No'

            b'Weak'
                b'Yes'

    Sunny
        Humidity
            b'High'
                b'No'

            b'Normal'
                b'Yes'
```