Program 4.

*Write a program to demonstrate the working of the decision tree based **ID3 algorithm**. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.*

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

class Node:
	def __init__(self, l):
		self.label = l
		self.branch = {}

def entropy(data):

	total_ex = len(data)
	p_ex = len(data.loc[data['PlayTennis']=='Yes'])
	n_ex = len(data.loc[data['PlayTennis']=='No'])
	en = 0
	if(p_ex>0):
		en = -(p_ex/float(total_ex)) * (math.log(p_ex,2)-math.log(total_ex,2))
	if(n_ex>0):
		en += -(n_ex/float(total_ex)) * (math.log(n_ex,2)-math.log(total_ex,2))

	return en

def gain(en_s,data_s,attrib):

	values = set(data_s[attrib])
	#print(values)
	gain = en_s
	for value in values:
		gain -= len(data_s.loc[data_s[attrib]==value])/float(len(data_s)) *
entropy(data_s.loc[data_s[attrib]==value])

	return gain

def get_attr(data):

	en_s = entropy(data)
	attribute = ""
	max_gain = 0

	for attr in data.columns[:len(data.columns)-1]:
		g = gain(en_s, data, attr)
		if g > max_gain:
```

```
                        max_gain = g
                        attribute = attr

        return attribute

def decision_tree(data):
        root = Node("NULL")
        if(entropy(data)==0):
                if(len(data.loc[data[data.columns[-1]]=="Yes"]) == len(data)):
                        root.label = "Yes"
                else:
                        root.label = "No"
                return root

        if(len(data.columns)==1):
                return
        else:
                attr = get_attr(data)
                root.label = attr

                values = set(data[attr])

                for value in values:
                        root.branch[value] =
decision_tree(data.loc[data[attr]==value].drop(attr,axis=1))

                return root

def get_rules(root, rule, rules):

        if not root.branch:
                rules.append(rule[:-1]+"=>"+root.label)
                return rules
        for val in root.branch:
                # print(val)
                get_rules(root.branch[val], rule+root.label+"="+str(val)+"^", rules)

        return rules

def test(tree, test_str):
        if not tree.branch:
                return tree.label
        return test(tree.branch[str(test_str[tree.label])], test_str)

data = pd.read_csv("tennis.csv")

tree = decision_tree(data)
rules = get_rules(tree," ",[])

for rule in rules:
```

```
        print(rule)

test_str = {}
print("Enter the test case input: ")

for attr in data.columns[:-1]:
        test_str[attr] = input(attr+": ")
print(test_str)

print(test(tree, test_str))
```