Code & OUTPUT

In [1]:

```
print("Experiment No 06 : To implement decision tree using ID3 algorithm.")
```

```
Experiment No 06 : To implement decision tree using ID3 algorithm.
```

In [4]:

```
import pandas as pd
import numpy as np
from collections import Counter

print("\tOUTPUT:\n\n")

# Function to calculate entropy of a dataset
def entropy(data):
    total_count = len(data)
    counts = Counter(data)
    ent = 0
    for count in counts.values():
        prob = count / total_count
        ent -= prob * np.log2(prob) if prob > 0 else 0
    return ent

# Function to calculate information gain for a feature
def information_gain(data, feature):
    total_entropy = entropy(data.iloc[:, -1])  # The target column is the last column
    feature_values = data[feature].unique()

    weighted_entropy = 0
    for value in feature_values:
        subset = data[data[feature] == value]
        weighted_entropy += (len(subset) / len(data)) * entropy(subset.iloc[:, -1])

    return total_entropy - weighted_entropy

# Function to build the decision tree using ID3
def id3(data, features):
    # Base case: If all rows have the same class, return the class label
    if len(set(data.iloc[:, -1])) == 1:
        return data.iloc[0, -1]

    # Base case: If no features left to split, return the majority class label
    if len(features) == 0:
        return Counter(data.iloc[:, -1]).most_common(1)[0][0]

    # Select the feature with the highest information gain
    gains = [information_gain(data, feature) for feature in features]
    best_feature = features[np.argmax(gains)]

    # Create a decision tree node
    tree = {best_feature: {}}
    remaining_features = [feature for feature in features if feature != best_feature]

    # Split the data based on the selected feature and recursively build subtrees
    for value in data[best_feature].unique():
        subset = data[data[best_feature] == value]
        subtree = id3(subset, remaining_features)
        tree[best_feature][value] = subtree

    return tree

# Function to make predictions using the decision tree
def predict(tree, instance):
    if isinstance(tree, dict):
        feature = list(tree.keys())[0]
        value = instance[feature]
        subtree = tree[feature].get(value)
        return predict(subtree, instance)
    else:
        return tree

# Example of how to use ID3 to classify the Iris dataset

from sklearn.datasets import load_iris
import pandas as pd

# Load Iris dataset and create DataFrame
iris = load_iris()
data = pd.DataFrame(iris.data, columns=iris.feature_names)
data['species'] = iris.target

# Convert the numeric target values to actual species names for better readability
data['species'] = data['species'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica'})

# Features to consider for splitting (excluding the target variable)
features = data.columns[:-1]  # Extract feature column names

# Build the decision tree using ID3
tree = id3(data, features)

# Print the decision tree
import pprint
pprint.pprint(tree)

# Making predictions on a new instance (just an example)
new_instance = data.iloc[0]  # You can choose any instance to test
prediction = predict(tree, new_instance)
print(f"\n\t Prediction for the first instance: {prediction}")
```

OUTPUT:

```
{'petal length (cm)': {1.0: 'setosa',
                       1.1: 'setosa',
                       1.2: 'setosa',
                       1.3: 'setosa',
                       1.4: 'setosa',
                       1.5: 'setosa',
                       1.6: 'setosa',
                       1.7: 'setosa',
                       1.9: 'setosa',
                       3.0: 'versicolor',
                       3.3: 'versicolor',
                       3.5: 'versicolor',
                       3.6: 'versicolor',
                       3.7: 'versicolor',
                       3.8: 'versicolor',
                       3.9: 'versicolor',
                       4.0: 'versicolor',
                       4.1: 'versicolor',
                       4.2: 'versicolor',
                       4.3: 'versicolor',
                       4.4: 'versicolor',
                       4.5: {'sepal length (cm)': {4.9: 'virginica',
                                                    5.4: 'versicolor',
                                                    5.6: 'versicolor',
                                                    5.7: 'versicolor',
                                                    6.0: 'versicolor',
                                                    6.2: 'versicolor',
                                                    6.4: 'versicolor'}},
                       4.6: 'versicolor',
                       4.7: 'versicolor',
                       4.8: {'sepal length (cm)': {5.9: 'versicolor',
                                                    6.0: 'virginica',
                                                    6.2: 'virginica',
                                                    6.8: 'versicolor'}},
                       4.9: {'sepal width (cm)': {2.5: 'versicolor',
                                                   2.7: 'virginica',
                                                   2.8: 'virginica',
                                                   3.0: 'virginica',
                                                   3.1: 'versicolor'}},
                       5.0: {'sepal length (cm)': {5.7: 'virginica',
                                                    6.0: 'virginica',
                                                    6.3: 'virginica',
                                                    6.7: 'versicolor'}},
                       5.1: {'sepal length (cm)': {5.8: 'virginica',
                                                    5.9: 'virginica',
                                                    6.0: 'versicolor',
                                                    6.3: 'virginica',
                                                    6.5: 'virginica',
                                                    6.9: 'virginica'}},
                       5.2: 'virginica',
                       5.3: 'virginica',
                       5.4: 'virginica',
                       5.5: 'virginica',
                       5.6: 'virginica',
                       5.7: 'virginica',
                       5.8: 'virginica',
                       5.9: 'virginica',
                       6.0: 'virginica',
                       6.1: 'virginica',
                       6.3: 'virginica',
                       6.4: 'virginica',
                       6.6: 'virginica',
                       6.7: 'virginica',
                       6.9: 'virginica'}}
```

```
Prediction for the first instance: setosa
```

In [ ]: