Decision Tree Problem

You may use C, C++, Java, C#, Python, or other languages to implement the algorithm. To simplify the implementation, your system only needs to handle **binary classification tasks** (i.e., each instance will have a class value of 0 or 1). In addition, you may assume that **all attributes have categorical values (not continuous)** and that there are **no missing values** in the training or test data. Sample training files (`train-*.dat`) and test files (`test-*.dat`) can be found from the course homework page.

The first line holds the attribute names, each name followed by an integer representing the number of possible values for that attribute. Each following line defines a single example. Each column holds this example's value for the attribute given at the top of the file (same order). The last column holds the class label for the examples. In all of the following experiments, you should use this last class attribute to help train the tree and to determine whether a tree classifies an example correctly.

In a decision tree, if you reach a leaf node but still have examples that belong to different classes, then choose the most frequent class (among the instances at the leaf node). If you reach a leaf node in the decision tree and have no examples left or the examples are equally split among multiple classes, then choose the class that is most frequent in the ***entire*** training set.

You do **not** need to implement pruning.

IMPORTANT: Your program should take only **two** arguments to be specified in the command line invocation of your program: a training file and a test file. There should be no graphical user interface (GUI). Any program that does not conform to the above specification will receive no credit. Also, don't forget to use logarithm base 2 when computing entropy and set $0 \log_2 0$ to 0.

(A). Build a decision tree using the training instances and print to `stdout` the tree in the same format as the example tree shown below (the symbol after colon denotes the class label for a leaf node).

```
attr1 = 0 :
| attr2 = 0 :
| | attr3 = 0 : 1
| | attr3 = 1 : 0
| attr3 = 1 :
| | attr4 = 0 : 0
| | attr4 = 1 : 1
attr1 = 1 :
|attr2 = 1 : 1
|
```

(B). Use the learned decision tree to classify the **training** instances. Print to `stdout` the accuracy of the tree. (In this case, the tree has been trained *and* tested on the same data set.) The accuracy should be computed as the percentage of examples that were correctly classified. For example, if 86 of 90 examples are classified correctly, then the accuracy of the decision tree would be 95.6%.

```
Accuracy on training set (90 instances): 95.6%
```

(C) Use the learned decision tree to classify the **test** instances. Print to `stdout` the accuracy of the tree. (In this case, the decision tree has been trained and tested on different data sets.)

```
Accuracy on test set (10 instances): 60.0%
```