```
Ryan Smith
CS 540: AI
HW #1

     COLOR        possible values:     Red, Green, Blue
     AGE          possible values:     Young, Old
     WEIGHT       possible values:     Light, Medium, Heavy

TRAIN set:
     COLOR = R    AGE = Y   WEIGHT = H    CATEGORY = +      (trnEx1)
     COLOR = G    AGE = O   WEIGHT = L    CATEGORY = +      (trnEx2)
     COLOR = R    AGE = Y   WEIGHT = H    CATEGORY = +      (trnEx3)
     COLOR = G    AGE = O   WEIGHT = H    CATEGORY = +      (trnEx4)
     COLOR = B    AGE = O   WEIGHT = H    CATEGORY = -      (trnEx5)
     COLOR = R    AGE = Y   WEIGHT = L    CATEGORY = -      (trnEx6)
     COLOR = G    AGE = Y   WEIGHT = L    CATEGORY = -      (trnEx7)

1.

a)

First we'll calculate the total info needed:

    InfoNeeded(f+, f-)  = -(f+)log(f+) - (f-)log(f-)

                        = -(4/7)log(4/7) - (3/7)log(3/7)

                        ≈ -(4/7)(-0.807354922057604) - (3/7)(-1.222392421336448)

                        = 0.4613456697472023 + 0.5238824662870491

                        = 0.9852281360342514

Next we'll calculate the remainders for each of the three features:

    Remainder(color)    = (3/7)I((2/3), (1/3)) + (3/7)I((2/3), (1/3)) + (1/7)I(0, 1)
    //red + green + blue

                        = (6/7)I((2/3), (1/3))

    Remainder(age)      = (4/7)I((2/4), (2/4)) + (3/7)I((2/3), (1/3))
    //young + old

                        = 4/7 + (3/7)I((2/3), (1/3))

    Remainder(weight)   = (3/7)I((2/3), (1/3)) + (3/7)I((2/3), (1/3)) + (1/7)I(0, 1)
    //light + medium + heavy

                        = (3/7)I((1/3), (2/3)) + (0/7)I(0, 0) + (4/7)I((3/4), (1/4))

                        = (3/7)I((1/3), (2/3)) + (4/7)I((3/4), (1/4))

----------
```

```
Now we calculate I((2/3), (1/3)) and I((3/4), (1/4)):

    I((2/3), (1/3))     = -(2/3)log(2/3) - (1/3)log(1/3)

                        ≈ -(2/3)(-0.584962500721156) -(1/3)(-1.5849625007211563)

                        = 0.3899750004807707 + 0.5283208335737187

                        = 0.9182958340544894

    I((3/4), (1/4))     = -(3/4)log(3/4) - (1/4)log(1/4)

                        ≈ -(3/4)(-0.415037499278844) - (1/4)(-2)

                        = 0.311278124459133 + 1/2

                        = 0.811278124459133

----------

Back to the feature calculations:

    Remainder(color)    = (6/7)(0.9182958340544894)

                        = 0.7871107149038481

    Remainder(age)      = 4/7 + (3/7)(0.9182958340544894)

                        = 0.9649839288804954

    Remainder(weight)   = (3/7)(0.9182958340544894) + (4/7)(0.811278124459133)

                        = 0.393555357451924 + 0.4635874996909331

                        = 0.8571428571428571

----------

So selecting color first gives us the most information:

    bestFeature = color

    leftFeatures = {age, weight}

    tree (so far):

        -------
        |color|
        -------
       /
     R/
     /
```

```
We make a recursive call to ID3({trnEx1, trnEx3, trnEx6}, {age, weight}):

    InfoNeeded((2/3), (1/3))     = -(2/3)log(2/3) - (1/3)log(1/3)

                                 ≈ 0.9182958340544894                        //from earlier
                                 calculation

Feature calculations:

    Remainder(age)               = (3/3)I((2/3), (1/3)) + (0/3)I(0, 0)

                                 ≈ 0.9182958340544894

    Remainder(weight)            = (2/3)I((2/2), (0/2)) + (0/3)I(0, 0) + (1/3)I((0/1), (1/1))

                                 = 0

----------

So we select weight next, and that gives us complete information for this tree node.

We assign a + label to red examples with medium weight even though there are none with medium
weight because most (2/3) red examples have + labels.

    tree (so far):

            -------
            |color|
            -------
           /    |
        R/    G|
        /      |
   -------
  |weight|
   -------
    /   |  \
  H/   M|   \L
  /     |    \
---    ---   ---
|+|    |+|   |-|
---    ---   ---

Now we make a recursive call to ID3({trnEx2, trnEx4, trnEx7}, {age, weight}):

    InfoNeeded((2/3), (1/3))     ≈ 0.9182958340544894                        //from earlier
    calculation

Feature calculations:

    Remainder(age)               = (1/3)I((0/1), (1/1)) + (2/3)I((2/2), (0/2))

                                 ≈ 0
    Remainder(weight)           = (2/3)I((1/2), (1/2)) + (0/3)I(0, 0) + (1/3)I((1/1), (0/1))

                                = 2/3

----------
```

So we select age next, and that gives us complete information for this tree node.

```
               -------
               |color|
               -------
              /    |
            R/    G|
            /      |
    --------     -----
    |weight|     |age|
    --------     -----
    /   |  \        |  \
  H/   M|   \L     Y|   \O
  /     |    \      |    \
---    ---   ---   ---   ---
|+|    |+|   |-|   |-|   |+|
---    ---   ---   ---   ---
```

Now we make a recursive call to ID3({trnEx5}, {age, weight}):

```
    InfoNeeded((0/1), (1/1))    = 0         //the one blue example has a - label
```

```
               -------
               |color|
               -------
              /    |  \
            R/    G|   \B
            /      |    \
    --------     -----   \
    |weight|     |age|    ---
    --------     -----    |-|
    /   |  \        |  \  ---
  H/   M|   \L     Y|   \O
  /     |    \      |    \
---    ---   ---   ---   ---
|+|    |+|   |-|   |-|   |+|
---    ---   ---   ---   ---
```

That completes our run of ID3 on our training data.

b)

```
TEST set:
     COLOR = G    AGE = O   WEIGHT = M    CATEGORY = +      (testEx1)
     COLOR = R    AGE = Y   WEIGHT = H    CATEGORY = +      (testEx2)
     COLOR = R    AGE = O   WEIGHT = H    CATEGORY = +      (testEx3)
     COLOR = G    AGE = Y   WEIGHT = H    CATEGORY = -      (testEx4)
     COLOR = B    AGE = O   WEIGHT = L    CATEGORY = -      (testEx5)
```

Applying our decision tree to the test data:

```
    testEx1: G -> O -> +    (correct)

    testEx2: R -> H -> +    (correct)

    testEx3: R -> H -> +    (correct)

    testEx4: G -> Y -> -    (correct)

    testEx5: B -> -         (correct)
```

Our tree yields 100% accuracy on the test results.

The test data is consistent with what's suggested by our training data. For example, we are led to believe that when an example is red, then weight is the best subsequent indicator of an example's label. If we looked at testEx2, for example, and went looked at age insead of weight next, we'd be incorrectly led to a selection of - as its label.

We have a couple data points that interesting to take note of - testEx1, for example, has a weight of M, which wasn't the case for any of the training examples. Our arc for the weight value of M takes us to + only because most red examples are + examples. This happens to be the case for testEx1, but this feels as though it could have been wrong because none of our traing data had M weight examples to work with.

It's a similar story for testEx5, which has color B. This makes us automatically assume that the example has a - label because 100% of our blue training examples have this label. Again, this happens to be correct for testEx5.

2.

TRAIN set:

```
     COLOR = R    AGE = Y   WEIGHT = H    CATEGORY = +      (trnEx1)
     COLOR = G    AGE = O   WEIGHT = L    CATEGORY = +      (trnEx2)
     COLOR = R    AGE = Y   WEIGHT = H    CATEGORY = +      (trnEx3)
     COLOR = G    AGE = O   WEIGHT = H    CATEGORY = +      (trnEx4)
     COLOR = B    AGE = O   WEIGHT = H    CATEGORY = -      (trnEx5)
     COLOR = R    AGE = Y   WEIGHT = L    CATEGORY = -      (trnEx6)
     COLOR = G    AGE = Y   WEIGHT = L    CATEGORY = -      (trnEx7)
```

TEST example:

```
    COLOR = B    AGE = O   WEIGHT = L    CATEGORY = -       (testEx5)
```

Applying the 1-nearest neighbor algorihtm is equivalent in this case to finding the closest 1 example in our training set to this example in the feature space and applying the label from that training example to this testing example.

Since this is a discrete feature space that's pretty small, we can check for examples in the training set one unit of distance at a time away from the testing set until we arrive at the closest.

```
Distance of 0?
    There are no examples in the training set with testEx5's exact feature values, so there is
    no example with a distance of 0 from testEx5.

Distance of 1?
    Starting with color, there's exactly one example from the training set with the same value
    (B) - trnEx5
        Then to age - trnEx5 has the same age value (O) as testEx5
            Then to weight - trnEx5 has a weight value of H, which is different from testEx5's
            value, which is L
                This example is a distance of 1 from testEx5, making it the nearest 1 neighbor
                to testEx5

We halt the algorithm here and apply trnEx5's label to testEx5. Our prediction for testEx5 is
therefore -, which is accurate.
```