Jerry Reptak & Jeremy Schiff
Algorithms Homework 10
4/26/2012

We are making a modified version of the leven

If a vowel is deleted, it costs 3 (boat -> bat)
If a vowel is inserted, it costs 3 (hill -> hilal)
If a vowel is changed to another vowel, it costs 2 (hot -> hit)
If a consonant is deleted, it costs 2 (trace -> race)
If a consonant is inserted, it costs 2 (rain -> ratin)
If a consonant is changed to another consonant, it costs 1 (shore -> snore)
(a consonant cannot be changed to a vowel or vice versa)

To change a consonant to a vowel has a cost of 5 (Delete consonant 2, insert vowel 3)
To change a vowel to a consonant has a cost of 5 (Delete vowel 3, insert consonant 2)

```
int LevenshteinDistance(char s[1..m], char t[1..n])
{
  // for all i and j, d[i,j] will hold the Levenshtein distance between
  // the first i characters of s and the first j characters of t;
  // note that d has (m+1)x(n+1) values
  declare int d[0..m, 0..n]

  d[0, 0] = 0

   // the distance of any first string to an empty second string
  for i from 1 to m
    if( isVowel(s[i]))
          d[i, 0] = d[i - 1, 0] + 3
    else
          d[i,0] = d[i - 1, 0] + 2

// the distance of an empty  first string to any second string
  for j from 1 to n
    if( isVowel(t[j]))
          d[0, j] = d[0, j - 1] + 2
    else
          d[0, j] = d[0, j - 1] + 3


  for j from 1 to n
  {
    for i from 1 to m
    {
      if s[i] = t[j] then
          d[i, j] := d[i-1, j-1]       // no operation required
      else
```

```
        if (isVowel(s[i]))
            // Vowel in our source
            d[i, j] = minimum
            (
                        d[i-1, j] + 3,  // a deletion
                          d[i, j-1] + 3,  // an insertion
                          d[i-1, j-1] + isVowel(t[j] ? 2 : 5  // a substitution, 2 if vowel, 5 if consonant
            )

        else {
            // consonant in our source

            d[i, j] = minimum
            (
                        d[i-1, j] + 2,  // a deletion
                          d[i, j-1] + 2,  // an insertion
                          d[i-1, j-1] + isVowel(t[j] ? 5 : 1  // a substitution, 5 if vowel, 1 if consonant
            )
        }
    }

    return d[m,n]
}
```