Arrays of Strings

If a string is a one-dimensional array, an array of strings is really a two-dimensional array. Here is an example definition:

```
char words[100][20];
```

This allocates 100 arrays of size 20, NOT 20 arrays of size 100.

Thus, we can store up to 100 strings, each of which can be no longer than 19 characters.

To access a particular string, only use one array index:

```
strcpy(words[0],"hello");
```

Here is an example that should illustrate the different expressions we can use when dealing with an array of strings:

```
char words[100][20];
int freq[26], i, wordcnt, let;
// Assume that the array gets filled here.

for (i=0; i<26; i++)
  freq[i] = 0;

for (wordcnt=0; wordcnt<100; wordcnt++) {

  for (let=0;let<strlen(words[wordcnt]);let++) {

    if (isalpha(words[wordcnt][let]))
      freq[tolower(words[wordcnt][let])-'a']++;
  }
}
```

Basically, we keep a frequency count of all alphabetic characters in the 100 strings in the array.

Arrays of Strings as Parameters

Consider turning the segment of code above into a function. We would want to pass in the two dimensional array as a parameter. Also, to make this a little easier, we would want to pass in the zeroed frequency array in as a parameter as well. One piece of information we DO have to specify in the function definition and prototype is the size of the second dimension.

```
void calcfreq(char words[][20], int freq[], int len)
{

  int wordcnt, let;

  for (wordcnt=0; wordcnt<len; wordcnt++) {

    for (let=0;let<strlen(words[wordcnt]);let++) {

      if (isalpha(words[wordcnt][let]))
        freq[tolower(words[wordcnt][let])-'a']++;
    }
  }
}
```

And here is how the function would be called:

```
calcfreq(words, freq, 100);
```

Sorting Arrays of Strings

We have no need to change our sorting algorithm when sorting strings. However, we must remember to properly use string functions as necessary. When sorting, we must do the following things:

1) Compare to items to see which one is "bigger".

2) Swap items in an array, which involves copying the contents of one string variable into another one.

Normally, with integers, we do the things above with operators, such as > and =. But, these won't properly work with strings. Instead, we must use strcmp and strcpy, respectively.

In the posted class example, an insertion sort is used. Note that we simply refer to a single string in a 2D character array by using one array index. The whole code is included below, pay careful attention to the use of strcmp and strcpy:

```
void sort(char words[][MAXLEN], int length) {

  int i, j;
  char temp[MAXLEN];

  // Loop through each element, inserting it into its proper location.
  for (i=1; i<length; i++) {

    j = i;

    // Continue swapping elements as long as the current element is not
    // in the right location. Note the use of strcpy.
    while (j>0 && strcmp(words[j], words[j-1]) < 0) {
        strcpy(temp, words[j]);
        strcpy(words[j], words[j-1]);
        strcpy(words[j-1], temp);
        j--;
    }
  }
}
```