Intro to Computing Lab 9 October 21, 2009

Objectives: You will gain experience using C++:

- defining functions
- arrays
- parameter passing of arrays

Download the following file to your desktop: http://www.cs.uni.edu/~fienup/cs051f09/labs/lab9.zip
Extract this file by right-clicking on lab9.zip icon and selecting Extract All.

The "Grade Book" example we have been following has a textual data file, `students.txt`, with the format shown below. Conceptually, this information can be visualized in the parallel arrays as:

```
3 5
Doe, Jane
1 2 3 4 5
Morse, Cody
1.1 2.2 3.3 4.4 5.5
Smith, John
10 20 30 40 50
HW 1
HW 2
Lab 1
Lab 2
Test 1
```

columnTitles:

0	1	2	3	4	5		49
HW 1	HW 2	Lab 1	Lab 2	Test 1			

studentNames:

0	"Doe, Jane"
1	"Morse, Cody"
2	"Smith, John"
3	
99	

scores:

	0	1	2	3	4	5		49
0	1.0	2.0	3.0	4.0	5.0			
1	1.1	2.2	3.3	4.4	5.5			
2	10.0	20.0	30.0	40.0	50.0			
3								
99								

Part A: Yesterday in lecture, we decided that it would be useful to have a function `listSearch`:

```
// Search the sorted stringList for the target. If the target is in the stringList, the function returns with found equal to
// true and locationFound specifying the index where target was found in stringList. If the target is not in the list, then the //
function returns with found equal false and locationFound specifying where the target should be added to the stringList.
void listSearch(char target[], char stringList[][NAME_SIZE], int stringCount, bool & found, int & locationFound);
```

The lab9.zip file you downloaded and extracted contains a `grade_book_starter` folder with a Visual Studio C++ project file: `parallelAnd2DArrays.sln` inside. Double-click on it to start this project in Visual Studio.

a) Your first task is to complete the function `listSearch` . Hints for implementing `listSearch`:

- use a loop to test items in stringList from index 0 upward until a match to the target is found, or an item greater than the target is found, or we have run out of items in the stringList
- use strcmp to compare two character arrays. Recall that strcmp(string1, string2) returns 0 if string1 is equal to string2. It returns a negative value if string1 is less than string2, and it returns a positive value if string1 is greater than string2.

You can test your function by running the grade book program and selecting the "`(C)hange a score`" menu option which utilizes the `listSearch` function twice: once to find the location of a specified student name in the `studentNames` array, and once to find the location of a specified column title in the `columnTitles` array.

After you have Part A complete and the "`(C)hange a score`" menu option working correctly, raise your hand and we'll check your work.

Part B: In lecture, we developed the function `makeRoomForNewStudent` to help solve the `"(N)ew student"` menu option.

```
// Slides the students and scores "down" one row from the startingRow to the endingRow.
// This makes room to add a new student name and their scores in the startingRow.
void makeRoomForNewStudent(char students[][NAME_SIZE], double scores[][MAX_SCORES],
                           int startingRow, int endingRow, int scoreCount) {
      int row;

      for (row = endingRow; row >= startingRow; row--) {
            strcpy_s(students[row+1], NAME_SIZE, students[row]);
            scoreRowCpy(scores, row+1, row, scoreCount);
      } // end for
} // end makeRoomForNewStudent
```

The function `makeRoomForNewStudent` calls:

- the built-in C-string function strcpy(string1, string2) which copies string2 to string1 (I actually used the safer function strcpy_s that specifies the size of the receiving array)
- the user defined function `scoreRowCpy` to copy a whole row of scores to another row

Your task for Part B is to complet the function `scoreRowCpy`:

```
// Copies the sourceRowNumber row of scores to the destinationRowNumber row of scores
void scoreRowCpy(double scores[][MAX_SCORES], int destinationRowNumber, int sourceRowNumber, int scoreCount)
```

You can test your function by running the grade book program and selecting the `"(N)ew student"` menu option.

After you have Part B complete and the `"(N)ew student"` menu option working correctly, raise your hand and we'll check your work.

Nothing needs to turned in for this lab. Make sure that you log off the computer before you leave.

EXTRA CREDIT:
If you have time to kill and/or want some extra credit, complete the functions `makeRoomForNewColumnTitle` and `scoreColumnCpy` used to implement the "`(A)dd column of scores`" menu option.