Intro to Computing Lab 11 November 4, 2009

Objectives: You will gain experience using C++:

- pointer usage
- relationship between arrays and pointers
- array sorting using the idea of "walking pointers"
- dynamically allocated arrays

Download the following file to your desktop: http://www.cs.uni.edu/~fienup/cs051f09/labs/lab11.zip
Extract this file by right-clicking on lab11.zip icon and selecting Extract All.

Part A: Yesterday in class we converted the bubble sort code using indexing into the bubble sort using pointers.

```
void bubbleSort(int array[], int size) {  // bubble sort using indexing
   bool swap;
   int temp;
   int lastUnsorted;

   lastUnsorted = size-1;
   do {
      swap = false;

      for (int count = 0; count < lastUnsorted; count++) {

         if (array[count] > array[count + 1]) {
            temp = array[count];
            array[count] = array[count + 1];
            array[count + 1] = temp;
            swap = true;
         } // end if
      } // end for

      lastUnsorted--;
   } while (swap);
} // end bubbleSort
```

Bubble sort code using pointers:

```
void bubbleSort(int array[], int size) {  // bubble sort using pointers
  bool swap;
   int temp;
   int * ptrToTest;
   int * lastUnsorted;

   lastUnsorted = &(array[size-1]);
   do {
      swap = false;

      for (ptrToTest = array; ptrToTest < lastUnsorted; ptrToTest++) {

         if (*ptrToTest > *(ptrToTest+1)) {
            temp = *ptrToTest;
            *ptrToTest = *(ptrToTest+1);
            *(ptrToTest+1)= temp;
            swap = true;
         } // end if
      } // end for

      lastUnsorted--;
   } while (swap);
} // end bubbleSort
```

The lab11.zip file you downloaded and extracted contains a `selectionSort` folder with a Visual Studio C++ project file: `selectionSort.sln` inside. Double-click on it to open this project in Visual Studio.
Modify the `selectionSort` function to use pointers instead of indexing to improve its performance.

After you have debugged your selection sort code, raise your hand and demonstrate your program.

Part B: Yesterday in lecture we looked at *dynamically allocating memory* for an array from the "*heap*" using the "`new`" operator. The `new` operator returns a pointer to the first element in the dynamically allocated array. For example,

```
int * ptrToNewArray;          // does not point anywhere initially

ptrToNewArray = new int [100]; // allocates an arrary of 100 int's
```

We also discussed dynamically allocating rows in the `scores` array of the grade book program only as they are needed. Dynamically allocating rows has the advantages of:

- saving memory space since unused rows do not require space
- we can make room for a new student's scores by manipulating pointers and not copying whole rows

The lab11.zip file you downloaded and extracted contains a `grade_book_starter_lab11` folder with a project file: `parallelAnd2DArrays.sln` inside. To dynamically allocate rows of scores you'll need to change:

- the declaration of the scores array from:

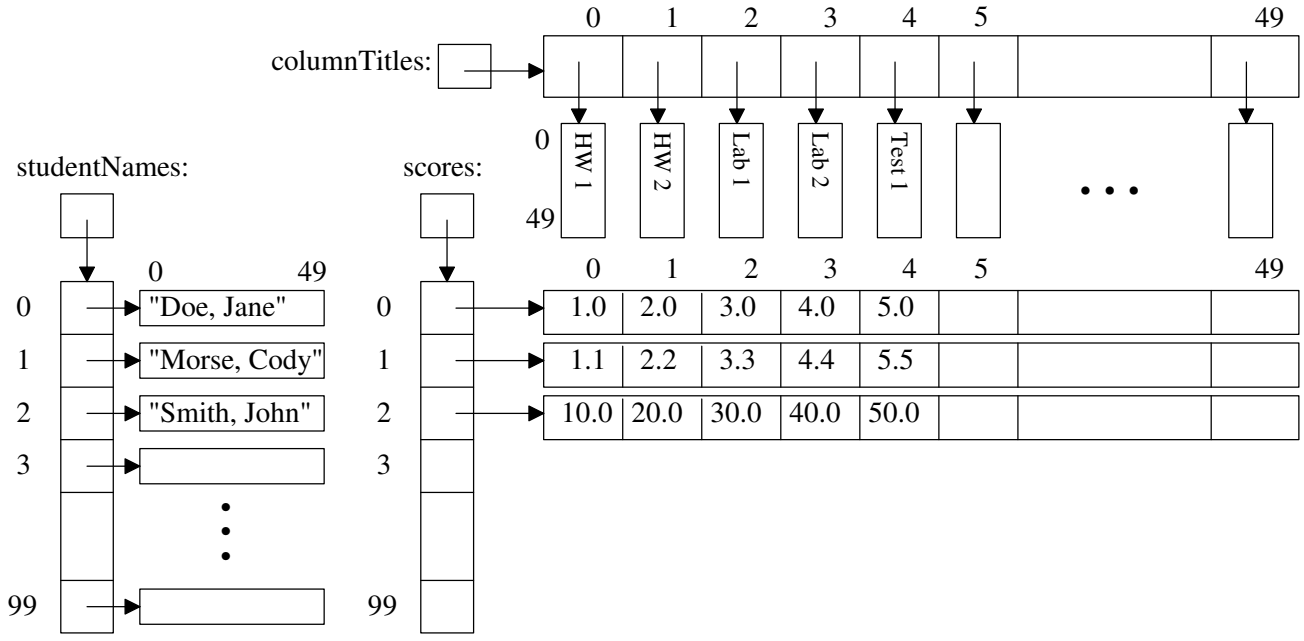
  ```
  double scores[CLASS_SIZE][MAX_SCORES];
  ```

  to

  ```
  double * scores[CLASS_SIZE]; // an array containing pointers to doubles
  ```

- all the prototype and function definitions to match, but this can be done using Edit | Find and Replace in the menu bar, i.e., Find "`double scores[][MAX_SCORES]`" and Replace with "`double * scores[]`"
- the `readScoresFromFile` function to dynamically allocate rows to hold the scores for each student
- the `scoreRowCpy` function to manipulate pointers to move a row of scores
- the `newStudent` function to dynamically allocate a row to hold the scores for the new student

There is a file `students.txt` available for testing. After you complete the above modifications, the resulting parallel arrays for this file are shown below. Notice that `scores` only has rows for existing students.



After you have completed and tested the above modifications, raise your hand demonstrate your program.

If you do not get done today, you can show me the completed lab in next week's lab period.

EXTRA CREDIT:
Modify the grade book program to eliminate wasted space associated with the `studentNames` and `columnTitles` arrays.