Pair Programming 7 Activities

* **Read the TESTS document first and then again after reading this document.**
* **Always use the pair programming tests to ensure your program works properly.**
* **Take a screen shot of each execution in the tests.**
* **Download the source code file for inclusion in the turn in document.**
* **Turn in pair programming activities using the pair programming turn in document.**
* **It is each individual’s responsibility to turn in the assignment and pair programming is graded individually so make sure you share the work you and your partner did together as you go. Do NOT share work that you don’t review together. It’s time to stop spoon feeding and/or leaching off of your partner(s).**
* **Do not share work with your partner that you did not do together.**
* **Make sure you have your partner’s name, username and contact information such as Pellissippi WebMail.**

7a:(1.5 points) Write a C++ program in a file named pp7a.cpp that declares and initializes an array of MAX\_STUDENTS grades like this:   
double grades[MAX\_STUDENTS] = { 99.9, 65.7, 87.8, 96.1, 56.8 } ;  
Use a for loop to print the grades and sum them. After you exit the loop, calculate and print the average. Challenge: initialize the grades in a sorted order and calculate the median grade. (Wait to try this challenge after you have completed the remaining chapter 7 pair programming activities.) See tests for more details.

7b:(1.5 points) Write a C++ program in a file named pp7b.cpp that uses one for loop to assign powers of 2 to the elements in the array declared below such that powers[0] should hold a 1 (since 20 = 1), powers[1] should hold a 2, powers[2] should hold a 4, etc. up to powers[10]. After this for loop completes assigning values to the powers array, use a second for loop to print the values in the array.

int powers[11]; // Holds 11 elements from 20 through 210

7c. (1.5 points) In a file named pp7c.cpp, write a function called *printMoney* that has one parameter, a double, and it prints this parameter formatted with a dollar amount with $ and exactly 2 digits to the right of the decimal. Write a driver (*main* function) that declares an array of monetary amounts like this:  
double amounts[MAX\_AMOUNTS];  
and uses a while or do while loop to ask the user for monetary amounts and put these values into the array. Do not allow the user to enter more than MAX\_AMOUNTS numbers. This loop must count how many numbers are placed in the amounts array as the array may be partially filled. Use a -1 for the amount as a sentinel value that will stop the loop (keeping in mind that the loop should ALSO stop when MAX\_AMOUNTS elements have been received from the user). After using a do or do while loop to fill the array, write a for loop to call the *printMoney* function to print all of the actual values in the amounts array. Do NOT print unfilled array elements. The constant and function declarations are shown below.

const int MAX\_AMOUNTS = 10;

void printMoney( double m );

7d: (1.5 points) In a file named pp7d.cpp, write a function called *printMonies* that has two parameters: #1 an entire array of doubles, #2 a count of the valid elements in the array. It prints every valid array element formatted like a dollar amount with $ and exactly 2 digits to the right of the decimal **using a for loop**. Write a driver (*main* function) that declares an array of monetary amounts like this:  
 double amounts[MAX\_AMOUNTS];  
and uses a while or do while loop to ask the user for monetary amounts. Do not allow the user to enter more than MAX\_AMOUNTS numbers. This loop must count how many numbers are placed in the amounts array as the array may be partially filled. Use a -1 for the amount as a sentinel value that will stop the loop (keeping in mind that the loop should ALSO stop when MAX\_AMOUNTS elements have been received from the user). After this loop, write code to call the function *printMonies* to print out all the elements of the amounts array.

// declaration for printMonies

void printMonies( double mArray[] , int n); // n contains the count of the actual number of values entered

7e:(2 points) In a file named pp7e.cpp, write code in a function named *linearSearch* that has 3 parameters: an array of integers, the number of valid elements in the array, and a key value to look for in the array. The function searches the array for the key value. If found, it returns the index where the matching element was found. If the key value is NOT found, the function returns -1 (an invalid index). Write a *main* driver function that initializes an array of integers and calls linearSearch as shown below. Create a global NOT\_FOUND constant whose value is -1.

foundIndex = linearSearch( array, n, key );

if ( foundIndex == NOT\_FOUND ) cout << “Not found\n”;

else cout << “Found element “ << array[foundIndex] <<

“ at index “ << foundIndex << endl;

7f: (2 points) In a C++ file named pp7f.cpp, write a function named *selectionSort* that has two parameters, an array of integers and the number of valid elements in the array. The code in this function follows the selection sort algorithm to sort the integers from lowest to highest, in ascending order. Write a *main* driver function that initializes an array with unsorted values, calls the *selectionSort* function, then prints the array values. **In the Week 9 content area, there is a link to a video that explains the selection sort.**