Pair Programming 7 Turn In

Name: \_\_\_\_\_Matthew Krahel \_\_\_\_\_\_\_\_\_ Username: \_\_\_\_\_\_\_C1010B11\_\_\_\_\_\_\_\_\_

Partner name: \_\_\_\_\_\_\_Blake Hodges\_\_\_\_\_\_\_\_\_\_\_ Partner username: \_\_\_\_C1010B06\_\_\_\_\_\_\_\_

\_X\_ I certify that my partner did work with me on these pair programming activities.

SCORE: \_\_\_\_\_\_\_\_\_\_\_\_ (to be filled in by instructor)

7a (1.5 points)

**/\* File: pp7a.cpp**

**\* Author: Blake Hodges and Matthew Krahel.**

**\* This program declares and initializes an array of MAX\_STUDENTS**

**\* Uses a for loop to print the grades and sum them.**

**\* Calculate and print the average.**

**\* Challenge: initialize the grades in a sorted order and**

**\* calculate the median grade.**

**\*/**

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**int main() {**

**//Declare Variables**

**const int MAX\_STUDENTS = 5;**

**double grades[MAX\_STUDENTS] = {99.9, 65.7, 87.8, 96.1, 56.8};**

**double totalGrade = 0;**

**double gradeAverage = 0;**

**int count = 0;**

**int medianArrayElem; //element number of the median**

**double medianNum; //Median number**

**//For Loop for printing the array elements**

**for (count = 0; count < MAX\_STUDENTS; count++)**

**{**

**//Display array element**

**cout << "The entered grades are: " << grades[count] << endl;**

**totalGrade += grades[count];**

**}**

**//Calculate average grade**

**gradeAverage = totalGrade/count;**

**//Display average grade**

**cout << "The Average for the class is: ";**

**cout << gradeAverage << endl;**

**//Display Median**

**cout << "The Median for the class is: ";**

**if(MAX\_STUDENTS % 2 == 0){ //if true then MaxStudents is even**

**medianArrayElem = MAX\_STUDENTS / 2;**

**medianNum = (grades[medianArrayElem - 1] + grades[medianArrayElem]) / 2;**

**}**

**else{**

**medianArrayElem = (MAX\_STUDENTS / 2); //should take only when MaxStudents is odd**

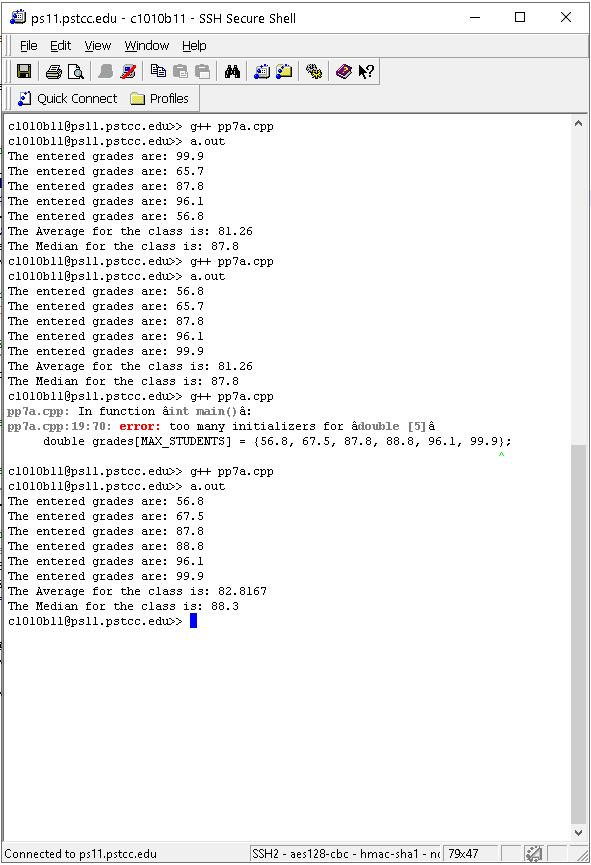
**medianNum = grades[medianArrayElem];**

**}**

**cout << medianNum << endl;**

**return 0;**

**}**



7b (1.5 points)

**/\* File: pp7b.cpp**

**\* Author: Blake Hodges and Matthew Krahel.**

**\* This program uses one for loop to assign powers of 2 to**

**\* the elements in the array and uses a for loop to print**

**\* the values in the array.**

**\*/**

**#include <iostream>**

**#include <fstream>**

**#include <cmath>**

**using namespace std;**

**int main()**

**{**

**//Declare Variables**

**const int NUM\_BASES = 11; // Max number of elements/Bases**

**int Powers2[NUM\_BASES]; // Array of powers of 2**

**// Input the powers for the array**

**for (int count = 0; count < NUM\_BASES; count++)**

**{**

**Powers2[count] = pow(2, count);**

**}**

**// Output the Array (powers of 2)**

**for (int count = 0; count < NUM\_BASES; count++)**

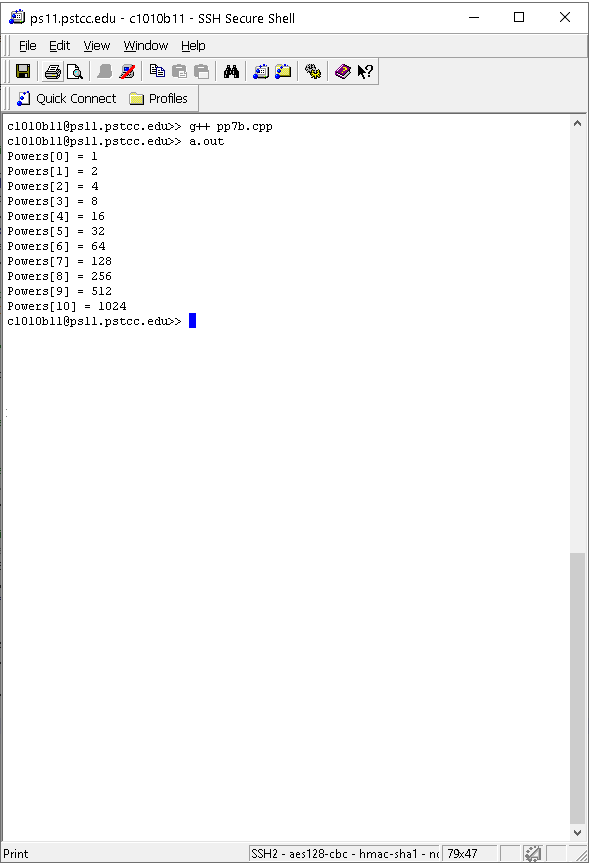
**{**

**cout << "Powers[" << count << "] = " << Powers2[count] << endl;**

**}**

**return 0;**

**}**



7c (1.5 points)

**/\* File: pp7c.cpp**

**\* Author: Blake Hodges and Matthew Krahel.**

**\* This program takes numbers from user, puts them in an**

**\* array, calls a function to output the elements in the**

**\* array formatted as dollars and cents.**

**\*/**

**#include <iostream>**

**using namespace std;**

**//declare functions**

**//precondition:function is passed the amounts array.**

**//postcondition:prints array elements in array.**

**void printMoney(double m[]);**

**int count = 0;**

**int main()**

**{**

**const int MAX\_AMOUNTS = 10;**

**double amounts[MAX\_AMOUNTS];**

**double amount= 0;**

**// Request numbers from user for the array**

**cout << "Enter number (-1 to end): ";**

**//Read in amounts**

**cin >> amount;**

**//Check array parameters**

**while (count < MAX\_AMOUNTS && amount != -1){**

**//Check valid entry or user quit**

**if (amount >= 0){**

**amounts[count] = amount;**

**count ++;**

**}**

**//check array size**

**if (count < MAX\_AMOUNTS){**

**cout << "Enter number (-1 to end): ";**

**cin >> amount;**

**}**

**}**

**//call function**

**printMoney(amounts);**

**return 0;**

**}**

**void printMoney(double m[])**

**{**

**cout.setf(ios::fixed);**

**cout.setf(ios::showpoint);**

**cout.precision(2);**

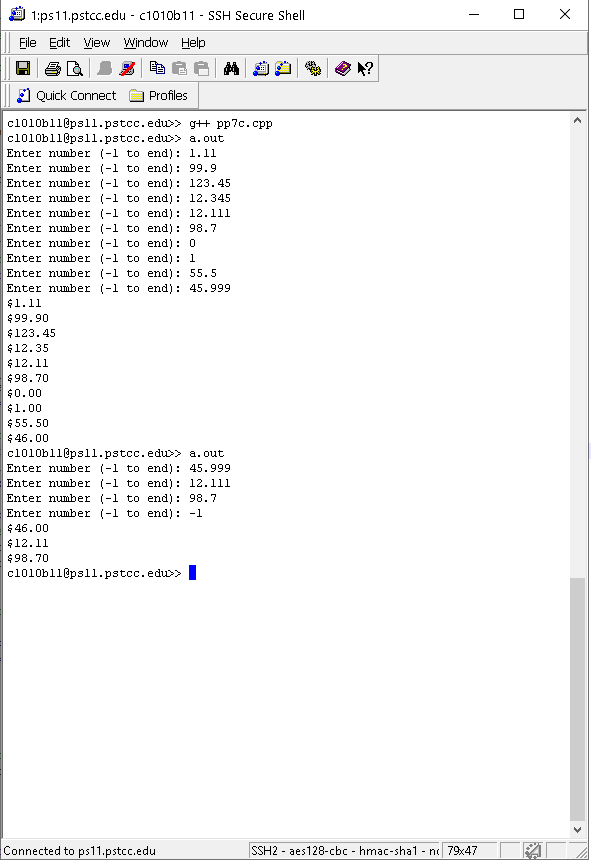
**for(int i = 0; i < count; i++)**

**{**

**cout << "$" << m[i] << endl;**

**}**

**}**



7d (1.5 points)

**/\* File: pp7d.cpp**

**\* Author: Blake Hodges and Matthew Krahel.**

**\* This program takes numbers from user, puts them in an**

**\* array, calls a function to output the elements in the**

**\* array formatted as dollars and cents.**

**\*/**

**#include <iostream>**

**using namespace std;**

**//declare functions**

**void printMonies(double mArray[], int n);**

**//precondition: an array of doubles to be formatted as dollars**

**//and cents, a count of the number of elements in the array.**

**//postcondition: output of the numbers in the array that are**

**//formatted as dollars and cents**

**int main()**

**{**

**//Declare Variables**

**const int MAX\_AMOUNTS = 10; //Max number of elements in amounts array**

**double amounts[MAX\_AMOUNTS]; // Array of numbers from user**

**int numElements; // Elements in amounts array**

**double userCancel = 1; // Used for if user doesn't need more elements**

**// Request numbers from user for the array**

**for (int i = 0; i < MAX\_AMOUNTS && userCancel >= 0 ; i++)**

**{**

**cout << "Enter number (-1 to end): ";**

**cin >> amounts[i];**

**numElements = i;**

**userCancel = amounts[i];**

**}**

**//Check for user cancelled**

**if(userCancel < 0){**

**numElements = numElements - 1;**

**}**

**// Call printMonies function**

**printMonies(amounts, numElements);**

**return 0;**

**}**

**void printMonies(double mArray[], int n)**

**//precondition: an array of doubles to be formatted as dollars**

**//and cents, a count of the number of elements in the array.**

**//postcondition: output of the numbers in the array that are**

**//formatted as dollars and cents**

**{**

**for(int i = 0; i <= n; i++)**

**{**

**//To display display 2 decimals**

**cout.setf(ios::fixed);**

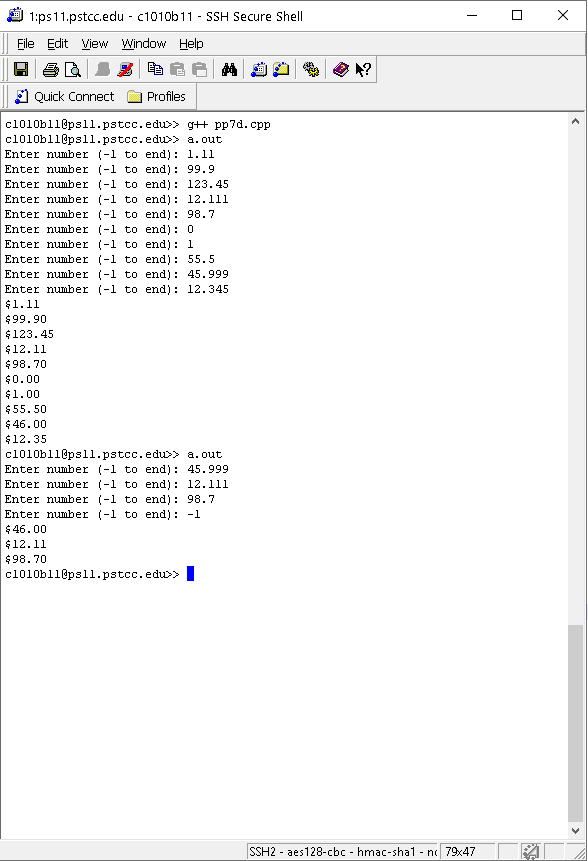
**cout.setf(ios::showpoint);**

**cout.precision(2);**

**cout << "$" << mArray[i] << endl;**

**}**

**}**



7e (2 points)

**/\* File: pp7e.cpp**

**\* Author: Blake Hodges and Matthew Krahel.**

**\* This program searches an array for a key element and returns**

**\* found element or did not find element**

**\*/**

**#include <iostream>**

**using namespace std;**

**//delcare global Variables**

**const int NOT\_FOUND = -1;**

**//declare functions**

**int linearSearch(const int array[], int n, int key);**

**//precondition: n is <= MAX\_SIZE, also array[] have values**

**//in index 0-MAX\_SIZE**

**//postcondition: returns int for element found that matches key**

**//or NOT\_FOUND if not element mathces key**

**int main(){**

**//declare function variables**

**int foundIndex = 0; // index returned from search**

**const int MAX\_SIZE = 8; // max size of array1**

**int key = 0; // search element**

**//Prompt and read in element to search for**

**cout << "Enter key value to search: " << endl;**

**cin >> key;**

**//Fill array**

**int array1[MAX\_SIZE] = {12, 1, 0, -5, 6, 8, 10, 18};**

**//set foundIndex variable with search function return**

**foundIndex = linearSearch(array1, MAX\_SIZE, key);**

**// check for index not found**

**if(foundIndex == NOT\_FOUND){**

**cout << "Not found!\n";**

**}**

**// if found display element location by index**

**else{**

**cout << "Found element " << array1[foundIndex] <<**

**" at index " << foundIndex << endl;**

**}**

**return 0;**

**}**

**int linearSearch(const int array[], int n, int key){**

**//declare function variables**

**int index = 0;**

**bool found = false;**

**//while loop if found is false and index is less than MAX\_SIZE**

**while ((!found) && (index < n)){**

**//search matches array**

**if (key == array[index]){**

**found = true;**

**}**

**//increment index if match not found**

**else {**

**index++;**

**}**

**}**

**//when found = true return index number for search**

**if (found){**

**return index;**

**}**

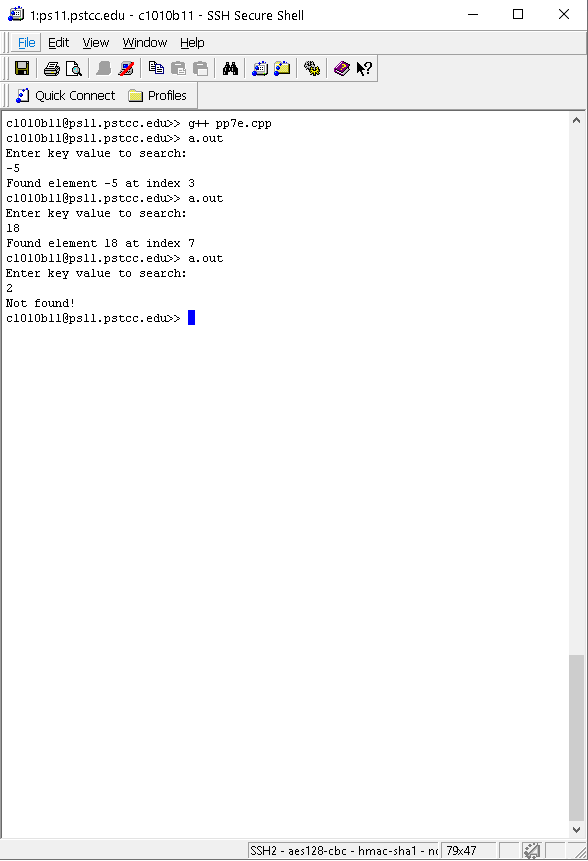
**//if no matches were found return -1**

**else{**

**return NOT\_FOUND;**

**}**

**}**



7f (2 points)

**/\* File: pp7.cpp**

**\* Author: Blake Hodges and Matthew Krahel.**

**\* This program initializes an array, calls the selection**

**\* sort function to sort the elements in ascending order,**

**\* then outputs the array.**

**\*/**

**#include <iostream>**

**using namespace std;**

**//declare functions**

**void selectionSort(int arrayToSort[], int count);**

**//precondition: an array of integers to be sorted in**

**//ascending order and the number of elements in the array**

**//postcondition: array is sorted by ascending order.**

**int main()**

**{**

**//Declare Variables**

**const int MAX\_AMOUNTS = 8; //Max number of elements in mArray**

**int mArray[MAX\_AMOUNTS] = { 12, 1, 0,**

**-5, 6, 8,**

**18, 10 }; // Array of numbers initialized**

**// Call selectionSort function**

**selectionSort(mArray, MAX\_AMOUNTS);**

**//Print array numbers (array values should be ascending order)**

**for(int i = 0; i < MAX\_AMOUNTS; i++)**

**{**

**cout << mArray[i] << " ";**

**}**

**//add line for output visual clarity**

**cout << endl;**

**return 0;**

**}**

**void selectionSort(int arrayToSort[], int count)**

**//precondition: an array of integers to be sorted in**

**//ascending order and the number of elements in the array**

**//postcondition: array is sorted by ascending order.**

**{**

**//Declare variables**

**int currentMin; //Current minimum element address**

**int tempElem; //Temporary placeholder for number in the minimum element address**

**//Sort the array in ascending order**

**for(int i = 0; i < count; i++)**

**{**

**//Initialize what element to start at**

**currentMin = i;**

**//Determine the minimum number in the remaining elements**

**for(int j = i+1; j < count; j++)**

**{**

**if(arrayToSort[j] < arrayToSort[currentMin])**

**{**

**currentMin = j;**

**}**

**}**

**//Take current minimum number and SWAP it with the current lowest element address**

**tempElem = arrayToSort[i]; //place number in current lowest element address in temp variables**

**arrayToSort[i] = arrayToSort[currentMin]; //place minimum number into current lowest element address**

**arrayToSort[currentMin] = tempElem; //place number from lowest element at start of loop into the location of the current minimum**

**}**

**}**

