**Lab Experience Twelve**

**Objectives:**

1. Performing a hand trace of array manipulation.

**Background**

Whenever a program is executed the memory is allocated for the variables declared within the program. All variables have two components associated with them: Address and contents. Therefore it is possible to change the contents of a variable just by referencing the variable’s address and not the name of the variable.

**Lab Exercises**

**Directions:**

This lab consists of writing computer programs using arrays, sorting and searching techniques. Answer the following exercises based on material presented in lecture and found in chapters 1-7 of the textbook.

**Exercise 1**

Answer the following questions by performing a hand-trace of the code by making a table of values using Microsoft Word’s table feature. **Write down the contents of the array after each pass of the outside *for loop*.**

NOTE: ***A screen shot of the output or just a listing of the output will be awarded a grade of zero.*** You need to perform a hand-trace, which means you are the computer. **See pages 130-132 for details about hand-tracing a program.**

1. Trace the following program.

const int SIZE = 6;

int main() {

int minIndex, minValue,

numValues = SIZE;

int a[SIZE] = {3, 2, 1, 5, 6, 4};

for (int j = 0; j < numValues - 1; j++) {

minIndex = j;

minValue = a[j];

for (int i = j + 1; i < numValues; i++)

if (a[i] < minValue) {

minValue = a[i];

minIndex = i;

}

a[minIndex] = a[j];

a[j] = minValue;

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| I | J | minIndex | minValue | a |
| 1 | 0 | 1 | 2 | {3,2,1,5,6,4} |
| 2 | 0 | 3 | 1 | (No change) |
| 3 | 0 | 3 | 1 | (No change) |
| 4 | 0 | 3 | 1 | (No change) |
| 5 | 0 | 3 | 1 | {1,2,3,5,6,4} |
| 2 | 1 | 2 | 3 | (No change) |
| 3 | 1 | 2 | 3 | (No change) |
| 4 | 1 | 2 | 3 | (No change) |
| 5 | 1 | 2 | 3 | {1,2,3,5,6,4} |
| 3 | 2 | 3 | 5 | (No change) |
| 4 | 2 | 3 | 5 | (No change) |
| 5 | 2 | 5 | 4 | {1,2,3,4,6,5} |
| 4 | 3 | 4 | 6 | (No change) |
| 5 | 3 | 5 | 5 | {1,2,3,4,5,6} |
| 5 | 4 | 5 | 6 | (No change) |

**Exercise 2:**

Do problem 12 on page 370 using the specifications specified in the problem. Copy and paste 3 individual runs of the console window showing the output into your word document. Make sure the first run shows that your data validation works more than once.

/\*

  Programmed by: Johnathan Lee

  Class: CSCI 1106

  Lab 12

  Takes input from 5 judges and displays the average after dropping the

  lowest and highest scores.

\*/

#include <iomanip>

#include <iostream>

**using** **namespace** std;

**const** **int** MAX\_SCORE = 10; // I doubt the lower bound would change,

// but the upper has a chance to.

**const** **int** NUM\_JUDGES = 5;

**void** getJudgeData(**double**& targetVar, **int** judgeNum);

**void** calcScore(**double** inScores[], **int** size);

**double** findLowest(**double** inScores[], **int** size);

**double** findHighest(**double** inScores[], **int** size);

**int** **main**() {

**double** scores[NUM\_JUDGES];

**for** (**int** i = 0; i < NUM\_JUDGES; i++) {

getJudgeData(scores[i], i);

}

calcScore(scores, NUM\_JUDGES);

**return** 0;

}

/\*

  getJudgeData()

  Gets a number from user, validates, and stores into a variable

  Pre: A reference to an integer.

  Post: targetVar has a score stored in it.

\*/

**void** getJudgeData(**double**& targetVar, **int** judgeNum) {

**cout** << "Please enter a score for judge #" << judgeNum + 1 << ": ";

**cin** >> targetVar;

// Only accept scores in [0, MAX\_SCORE]

**if** (targetVar < 0 || targetVar > MAX\_SCORE) {

**cout** << "Error: Score out of bounds [0," << MAX\_SCORE

<< "]. Terminating.\n";

**exit**(1106 - 12);

}

}

/\*

  calcScore()

  Given an array of scores, prints the average after dropping the highest and

  lowest.

  Pre: An array of 'size' with values in all.

  Post: Output

\*/

**void** calcScore(**double** inScores[], **int** size) {

**double** lowest, highest, sum = 0, avg;

**for** (**int** i = 0; i < NUM\_JUDGES; i++) {

sum += inScores[i];

}

lowest = findLowest(inScores, size);

highest = findHighest(inScores, size);

sum -= lowest + highest; // Drop the lowest and highest.

avg = sum / (NUM\_JUDGES - 2); // Sub 2 to account for lowest/highest dropped.

**cout** << fixed << showpoint << **setprecision**(2);

**cout** << "The average of the " << NUM\_JUDGES << " scores was: " << avg << **endl**

<< **endl**;

}

/\*

  findLowest()

  Finds and returns the lowest value in a given array of 'size'.

  Pre: An array of 'size' with values in all.

  Post: The lowest value of the array.

\*/

**double** findLowest(**double** inScores[], **int** size) {

**double** lowest = MAX\_SCORE;

**for** (**int** i = 0; i < size; i++) {

**if** (inScores[i] < lowest)

lowest = inScores[i];

}

**return** lowest;

}

/\*

  findHighest()

  Finds and returns the highest value in a given array of 'size'.

  Pre: An array of 'size' with values in all.

  Post: The highest value of the array.

\*/

**double** findHighest(**double** inScores[], **int** size) {

**double** highest = 0;

**for** (**int** i = 0; i < size; i++) {

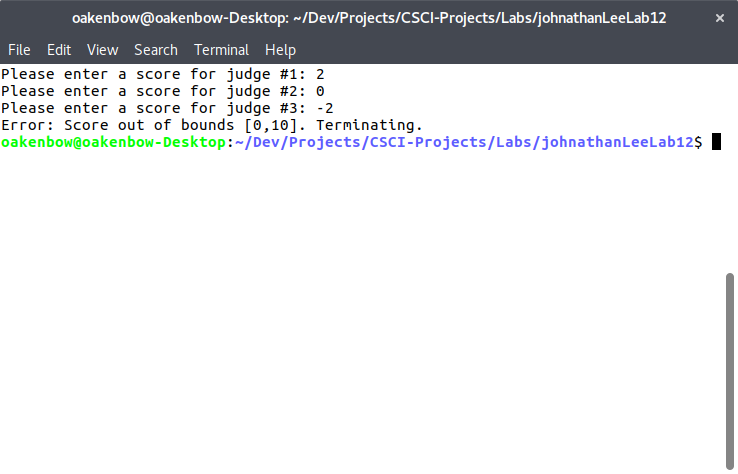
**if** (inScores[i] > highest)

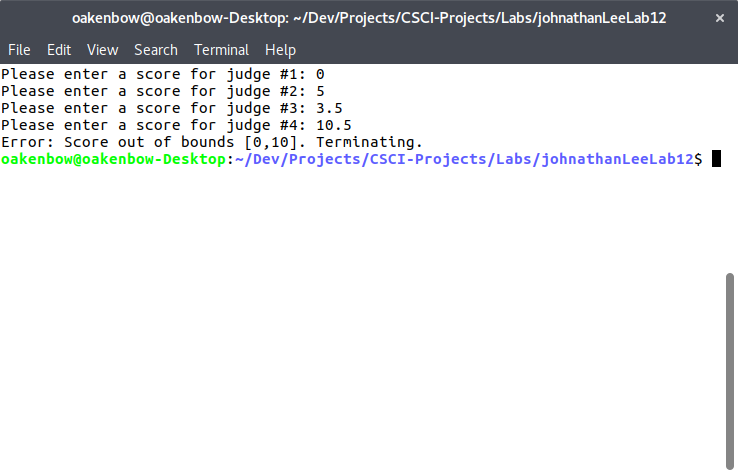
highest = inScores[i];

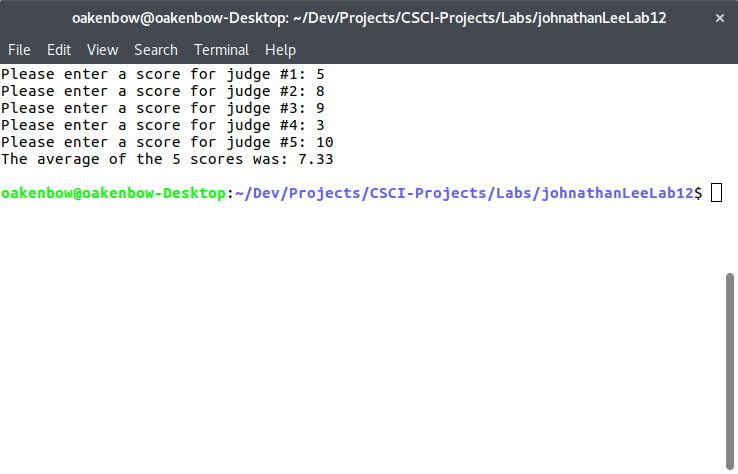
}

**return** highest;

}

****

****

****

**Due Dates:**

As indicated on the Lab Experience Twelve assignment folder.

**What to hand in:**

1. Hand in a print out of your word document and your program.
2. Compress your program the .cpp file and your word file into a single compressed folder called yournamelab12.zip.
3. Place your word document into the assignment Lab Experience Twelve folder.