CSCI 152 Programming Fundamentals II Spring 2018

Dr. Creider Program 4 – Computing Mode(s)

/\* CSCI 152 Spring 2018

Program 4 Your Name

Four short arrays with 1000 elements each – data array, numbers array and two other arrays

The user will enter a size value which is the number of values to be stored in the data set

The user will enter the data set to be stored in the data array

The user will enter a number which represents the frequency count of their interest

The program will display the data entered, compute a frequency distribution and find the unique mode if one exists,

find all values in the data set which occur with the frequency entered by the user and store the values in the

numbers array, and display the values stored in the numbers array

\*/

#include<iostream> // required header file for input/output

using namespace std;

//The program must include 5 functions

void inputData(short data[], short size); // function to enter data into the array

void displayData(short data[], short size); // function to display data in an array

void FrequencyDistribution(short data[], short size, short count[], short &sizeCU, /\*short unique[]\*/);

short FindMode(short numbers[], short &modeFrequency, /\* other parameters needed based on your algorithm\*/);

short SaveFreq (short frequency, short numbers[],/\* other parameters needed based on your algorithm\*/);

int main()

{

short data[1000], size, // array for original data to be processed

numbers[1000], // array to contain the actual numbers that had a specific frequency

sizeNumbers, // number of values store in numbers array

frequency, // frequency that user requested to identify in data set

count[1000], // array to be used in finding the frequency distribution

unique[1000], // array to be used in finding the frequency distribution if needed

sizeCU, // number of values stored in the count and unique arrays

modeFrequency, // the number of times (frequency) the mode(s) occurred

numModes; // number of modes found in the data set

cout<<"enter the number of values to store in the array or zero to terminate the program\n";

cin>>size;

while(size>0) // loop to permit user to test many data sets

{

cout<<"enter "<<size<<" numbers from the keyboard\n";

inputData(data, size);

// print the contents of the array

cout<<"there are "<<size<<" values in the array\n";

displayData(data, size);

FrequencyDistribution(data, size, count, sizeCU,/\*unique array if needed\*/);

numModes = FindMode(numbers, modeFrequency, /\* other arguments needed based on your algorithm\*/);

cout<<"There were "<<numModes<<" mode(s) in the data set, the mode frequency is "<<modeFrequency<<" the mode value(s) is/are:\n";

displayData(numbers, numModes);

cout<<"\nthe program will count the number of values that occurred with a specific frequency in the array\n";

cout<<"enter the frequency that you want to find or zero to stop finding frequencies\n";

cin>>frequency;

while(frequency>0) // loop to permit user to check many frequencies

{

sizeNumbers = SaveFreq (frequency, numbers /\* other parameters needed based on your algorithm\*/);

if(sizeNumbers) // if the numbers array contains any values, display the values stored in the array

{

cout<<sizeNumbers<<" values that occurred "<<frequency<<" time(s) were found in the "<<size<<" element array\n";

cout<<"the values are: ";

displayData(numbers, sizeNumbers);

}

else

cout<<"there were no values in the array that occurred "<<frequency<<" times\n";

cout<<"enter the frequency that you want to find or zero to stop finding frequencies\n";

cin>>frequency;

} // end of inner while loop

cout<<"enter the number of values to store in the array or zero to terminate the program\n";

cin>>size;

} // end of outer while loop

system("pause"); // pause the program to see the results

//return 0; // optional statement, may be required for .NET compiler

}

//function definitions are placed after main

// Function to display the data in the array, print all values on the same line with one space between each number

void display\_data(short data[], short size)

{ /\*code in function\*/ }

// **Do not output anything in these functions**

// Function to input(store) data into the array

void input\_data(short data[], short size)

{ /\*code in function\*/ }

// Function to compute a frequency distribution using one of the algorithms described below

void FrequencyDistribution(short data[], short size, short count[], short &sizeCU, /\*short unique[]\*/)

{ /\*code in function\*/ }

// Function to count the number of modes in the data set and return the count. Function will also store the actual mode

// value(s) in the numbers array.

short FindMode(short numbers[], short &modeFrequency, /\* other parameters needed based on your algorithm\*/)

{ /\*code in function\*/ }

// Function to return the count of the number of values which occurred exactly *frequency* times. The function will store in // the numbers array the specific numbers that occurred with the frequency value passed to the function.

short *SaveFreq* (short frequency, short numbers[],/\* other parameters needed based on your algorithm\*/)

{ /\*code in function\*/ }

Possible algorithms – (1) compute frequency distribution (use unique and count arrays); (2) use count array to store the frequency of a number

Assignment – Put in eCollege dropbox: Program 4

Due: 3/21/2018 before class

No global variables are permitted in this assignment. Write the missing C++ statements in the code above to complete the program. Compile and execute the program making sure that all syntax and logic errors are removed. Name the CPP file ‘LastNameFirstInitial-P4’. For example I would name my file CreiderD-p4.cpp. Upload only the cpp file for this assignment. Late submission forfeits 20% of grade and a grade of 0 will be given if assignment is submitted 7 or more days after it is due.

Write a value returning function, *SaveFreq*, which finds all values in the data set that have a frequency passed to the function and stores the values in a second array, *numbers*, passed to the function. Using a return statement, return the (size/count) number of values stored in the array *numbers*. You can use the code provided above to complete the assignment. The *SaveFreq* function call contains several arguments. You must pass the frequency the user entered and the array, numbers, in which to store the values that with the frequency the user entered. The other arguments will depend on the algorithm you used to compute the frequency distribution.

Example: the data set contains the following values: 1, 2, 3, 4, 5, 2, 1, 5, 3, 5

If the user enters 1 for a frequency you would store 4 in the numbers array and return a size of 1.

If the user enters 2 for a frequency you would store 1, 2, 3 in the numbers array and return a size of 3.

If the user enters 3 for a frequency you would store 5 in the numbers array and return a size of 1.

If the user enters any other frequency you would not store anything in the numbers array and return a value of 0.

The inputData and displayData functions are the same as the last program (except for the type of the arrays) and you can use the code from that program for this assignment. Two new functions you will write are the FrequencyDistribution function and the FindMode function. The FrequencyDistribution function will create a count of how many times each value in the array occurred. This can be done with one or two additional arrays in addition to the data array. I suggested two possible algorithms for you to use. The arguments to the function will be different depending on the algorithm you select to solve this problem.

The arguments for the FindMode function will depend on the algorithm used in the FrequencyDistribution function. At a minimum you must pass the array ***numbers***, and the variable ***modeFrequency*** by reference. The function will store the actual mode values in the array ***numbers*** and return with a return statement the number of modes found in the data set and stored in the array numbers. The variable ***modeFrequency*** will be assigned the frequency of the mode(S). See example below.

Example: the data set contains the following values: 1, 2, 3, 4, 5, 2, 1, 5, 3, 5

For this data set the mode is 5 and it occurred 3 times. The data set contains a unique mode so the function would return a 1 and 5 would be stored in the array numbers.

Example: the data set contains the following values: 1, 2, 3, 4, 5, 5, 4, 2, 1, 5, 3, 5, 4, 4

This data set contains a bi-modal distribution of 5 and 4 and they occurred 4 times, so the function would return a 2 and store the values 5 and 4 in the array numbers.

If all values in the array occurred with a frequency of one, the function would return the size of the array. For this case there actually is no mode but you do not have to test for that condition.