CS331
Program 1
James Francis
User 12

```
cat -b lab1.h
#ifndef LAB1_H
#define LAB1_H

#include <iostream>
#include <fstream>
#include <math.h>
using namespace std;

class lab1{

public:
    lab1();
bool checkValueTwo(int, int, int);
bool checkValueThree(int, int, int, int);
bool checkValueFour(int, int, int, int, int);
string swapNumforDarkChar(int);
};
#endif
:
printf \\n\\n


cat -b lab1.cpp
/*
 PROGRAM NAME: Program 1: Developing skills for 2d arrays in C++

 PROGRAMMER:   James Francis

 CLASS:        CSC 331.001, Fall 2014

 INSTRUCTOR:   Dr. Strader.

 DATE STARTED: September 9, 2014

 DUE DATE:     September 11, 2014

 PROGRAM PURPOSE:

 This program will read in values from a .dat file, store those values in a 2D array. It will then output
 that 2D array and a Dark Character representation of the input to the console.
 Then the program will check the nearest neighbor values (+/-)1 both laterally and vertically, if the
 current element differs from these values by more than one, the program will take the average of the
 neighbor values, rounded to the nearest integer and assign that value to the current element.

 Once this "error checking" has been completed, the Corrected Input will be printed to console along with
 a Dark Character representation of the Corrected Input.



 VARIABLE DICTIONARY:

int size: size of the 2 dimensional array, determined by the first line of input file
int left, right, top, bottom: values of the elements surrounding the current element
int intArray[][]: 2 dimensional array used to hold the input
string dashes, spaces: strings used solely for formatting, to make output look presentable(?)


 ADTs: none

 FILES USED:

 prog1.dat



 SAMPLE INPUTS:

 10
 7 6 9 4 5 4 3 2 1 0
 6 5 5 5 6 5 4 3 2 1
 6 5 6 6 7 6 8 4 3 2
 1 5 6 7 7 7 6 5 4 3
 5 5 6 7 6 7 7 6 5 9
 5 6 7 6 5 6 6 5 4 3
 5 6 7 9 5 5 6 5 4 3
 5 5 6 7 6 6 7 6 5 4
 5 9 5 6 7 6 5 0 3 2
 5 5 5 5 6 5 4 3 2 7



 SAMPLE OUTPUTS:

 Uncorrected Input
 ---------------------|
 7 6 9 4 5 4 3 2 1 0  |
 6 5 5 5 6 5 4 3 2 1  |
 6 5 6 6 7 6 8 4 3 2  |
 1 5 6 7 7 7 6 5 4 3  |
 5 5 6 7 6 7 7 6 5 9  |
 5 6 7 6 5 6 6 5 4 3  |
 5 6 7 9 5 5 6 5 4 3  |
 5 5 6 7 6 6 7 6 5 4  |
 5 9 5 6 7 6 5 0 3 2  |
 5 5 5 5 6 5 4 3 2 7  |
 |
 # * & ! + ! - , .    |
 * + + + * + ! - , .  |
 * + * * # * $ ! - ,  |
 . + * # # # * + ! -  |
 + + * # * # # * + &  |
 + * # * + * * + ! -  |
 + * # & + + * + ! -  |
 + + * # * * # * + !  |
 + & + * # * +   - ,  |
 + + + + * + ! - , #  |
 ---------------------|


 Corrected Input
 ---------------------|
 7 7 5 4 5 4 3 2 1 0  |
 6 6 5 5 6 5 5 3 2 1  |
 4 5 6 6 7 7 6 4 3 2  |
 5 5 6 7 7 7 6 5 4 5  |
 5 5 6 7 6 7 7 6 6 5  |
 5 6 7 7 6 6 6 5 5 4  |
 5 6 7 7 6 5 6 5 4 3  |
 5 7 6 7 6 6 6 4 4 3  |
 6 6 5 6 7 6 4 4 3 4  |
 5 5 5 5 6 5 4 3 4 4  |
 |
 # # + ! + ! - , .    |
 * * + + * + + - , .  |
 ! + * * # # * ! - ,  |
 + + * # # # * + ! +  |
 + + * # * # # * * +  |
 + * # # * * * + + !  |
 + * # # * + * + ! -  |
 + # * # * * * ! ! -  |
 * * + * # * ! ! - !  |
 + + + + * + ! - ! !  |
 ---------------------|

 ---------------------------------------------------------------------*/

#include "lab1.h"
```

```
lab1::lab1(){
    int size = 0;
    int left=0;
    int right=0;
    int bottom=0;
    int top = 0;

    //-----------------------------------------------------------------------------------------------------
    // Open input file and determine the size of the the array by assigning the first valeu in the file to
    // the size variable
    //-----------------------------------------------------------------------------------------------------

    ifstream infile("../instr/prog1.dat");
    for (int i=0;i<1;i++){
        infile>>size;
        cout<<endl;
    }

    int intArray[size][size];
    //--------------------------------------------------------------------------------------------
    // Populate the previously created intArray[][] of dimesions [size] x [size]
    //--------------------------------------------------------------------------------------------
    for (int i = 0; i<size; i++){
        for(int j = 0; j<size; j++){
            infile>>intArray[i][j];

        }
    }

    string spaces = "";
    string dashes = "-";

    //---------------------------------------------------------------------------------------------------
    // Begin segment of code to print uncorrected input and corresponding dark character for that input
    //---------------------------------------------------------------------------------------------------

    cout << "Uncorrected Input" <<std::endl;

    for (int i = 0; i <size;i++){
        dashes=dashes+"--";
        spaces=spaces+"  ";
    }

    cout << dashes << "| " <<std::endl;
    for (int i = 0; i < size; ++i){
        for (int j = 0; j < size; ++j){
            std::cout << intArray[i][j] << ' ';
        }
        std::cout << " | " <<std::endl;
    }

    cout << spaces <<" | " <<std::endl;

    for (int i = 0; i < size; ++i){
        for (int j = 0; j < size; ++j){
            int value = intArray[i][j];
            std::cout << lab1::swapNumforDarkChar(value) << ' ';
        }
        std::cout << " | " <<std::endl;
    }

    cout << dashes << "| " <<std::endl;
    cout <<std::endl<<std::endl;



    //---------------------------------------------------------------------------------------------
    // Begin segment of code to print Corrected Input and corresponding dark characters
    //---------------------------------------------------------------------------------------------

    cout << "Corrected Input" << std::endl;

    cout << dashes << "| " <<std::endl;
    int current = 0;
    for (int i = 0; i < size; i++){
        for (int j = 0; j < size; j++){
            current = 0;
            double avg=0;
            current = intArray[i][j];

            //---------------------------------------------------------------------------------------------
            //This statement handles the special case of the first element within the first row
            //---------------------------------------------------------------------------------------------
            if (i==0 && j==0) {
                //---------------------------------------------------------------------------------------------
                //This statement handles the special case of the first element within the first row
                //---------------------------------------------------------------------------------------------
                right = intArray[i][j+1];
                bottom = intArray[i+1][j];

                int difference1 = std::abs(current-bottom);
                int difference2 = std::abs(current-right);

                if((difference1==1 && difference2==1) || (difference1==0&&difference2 ==0) || (difference1==1 && difference2==0)|| (difference1==0 && difference2==1)){
                    intArray[i][j] = intArray[i][j];

                }
                else{
                    //--------------------------------------------------------------------------------------
                    // This statement is run if any of the difference values are more than 1
                    //--------------------------------------------------------------------------------------
                    avg = (right+bottom)/2.0;
                    intArray[i][j] = static_cast<int>(floor(avg+.5));
                }
            }


            //--------------------------------------------------------------------------------------
            //This statement handles the special case of the nonfirst and nonlast elements in the
            //first row
            //--------------------------------------------------------------------------------------
            if (i==0 && (j>0 && j<(size-1))) {
                //--------------------------------------------------------------------------------------
                //This statement handles the special case of the nonfirst and nonlast elements in the
                //first row
                //--------------------------------------------------------------------------------------

                left = intArray[i][j-1];


                right = intArray[i][j+1];

                bottom = intArray[i+1][j];

                if(lab1::checkValueThree(current,left, right, bottom)){
                    intArray[i][j] = intArray[i][j];

                }
                else{
                    //--------------------------------------------------------------------------------------
                    // This statement is run if checkValueThree is false
                    //--------------------------------------------------------------------------------------
                    avg =(left+right+bottom)/3.0;
                    intArray[i][j] = static_cast<int>(floor(avg+.5));

                }
            }

            //-----------------------------------------------------------------------------------------
            //This statement handles the special case of the last element within the first row
            //-----------------------------------------------------------------------------------------
            if (i==0 && j==size-1) {
                //-----------------------------------------------------------------------------------------
                //This statement handles the special case of the last element within the first row
                //-----------------------------------------------------------------------------------------

                left = intArray[i][j-1];
                bottom = intArray[i+1][j];
                if(lab1::checkValueTwo(current, left, bottom)==true){
                    intArray[i][j] = intArray[i][j];

                }
                else{
                    intArray[i][j] = static_cast<int>(floor(((left+bottom)/2.0)+.5));
                }
            }

            //-------------------------------------------------------------------------------------
            //This statement handles the special case of nonfirst and nonlast elements in the
            //first column
            //-------------------------------------------------------------------------------------
            if ( (i>0 && i<(size-1)) && j==0) {
                //-------------------------------------------------------------------------------------
                //This statement handles the special case of nonfirst and nonlast elements in the
                //first column
//-----------------------------------------------------------------------------

                top = intArray[i-1][j];
                right = intArray[i][j+1];

                bottom = intArray[i+1][j];

                if(lab1::checkValueThree(current,top, right, bottom)==true){
                    intArray[i][j] = intArray[i][j];

                }
                else{
         intArray[i][j] = static_cast<int>(floor(((top+right+bottom)/3.0)+.5));

                }
            }

                         //------------------------------------------------------------------
```

```
//This statement handles all elements not in the first or last row or column
         //-------------------------------------------------------------------------------------
         if ( (i>0 && i<(size-1)) && (j>0&&j<(size-1)) ){
             //-------------------------------------------------------------------------------------
             //This statement handles all elements not in the first or last row or column
             //-------------------------------------------------------------------------------------

             top = intArray[i-1][j];
             left = intArray[i][j-1];
             right = intArray[i][j+1];
             bottom = intArray[i+1][j];
             if(lab1::checkValueFour(current, top, left, right, bottom)){
                 intArray[i][j] = intArray[i][j];
             }
             else{
                 intArray[i][j] = static_cast<int>(floor(((top+left+right+bottom)/4.0)+.5));
             }
         }

         //-------------------------------------------------------------------------------------
         //This statement handles the special case of last elements in the last column
         //-------------------------------------------------------------------------------------
         if ( (i>0 && i<(size-1)) && j==(size-1)) {
             //-------------------------------------------------------------------------------------
             //This statement handles the special case of last elements in the last column
             //-------------------------------------------------------------------------------------

             top = intArray[i-1][j];
             left = intArray[i][j-1];
             bottom = intArray[i+1][j];
             if(lab1::checkValueThree(current,top, left, bottom)){
                 intArray[i][j] = intArray[i][j];
             }
             else{
                 intArray[i][j] = static_cast<int>(floor(((top+left+bottom)/3.0)+.5));
             }
         }

         //-----------------------------------------------------------------------------------------
         //This statement handles the special case of the first element within the last row
         //-----------------------------------------------------------------------------------------
         if (i==(size-1) && j==0) {
             //-----------------------------------------------------------------------------------------
             //This statement handles the special case of the first element within the last row
             //-----------------------------------------------------------------------------------------
             top = intArray[i-1][j];
```

```
                right = intArray[i][j+1];
                if(lab1::checkValueTwo(current, right, top)){
                    intArray[i][j] = intArray[i][j];
                }
                else{
                    intArray[i][j] = static_cast<int>(floor(((right+top)/2.0)+.5));
                }
            }

            //------------------------------------------------------------------------------------------
            //This statement handles the special case of the nonfirst and nonlast elements in the
            //last row
            //------------------------------------------------------------------------------------------
            if (i==(size-1) && (j>0&&j<(size-1))) {
                //------------------------------------------------------------------------------------------
                //This statement handles the special case of the nonfirst and nonlast elements in the
                //last row
                //------------------------------------------------------------------------------------------

                left = intArray[i][j-1];
                right = intArray[i][j+1];
                top = intArray[i-1][j];
                if(lab1::checkValueThree(current,left, right, top)){
                    intArray[i][j] = intArray[i][j];
                }
                else{
                    intArray[i][j] = static_cast<int>(floor(((left+right+top)/3.0)+.5));
                }
            }

            //------------------------------------------------------------------------------------------
            //This statement handles the special case of the first element within the first row
            //------------------------------------------------------------------------------------------
            if (i==size-1&&j==size-1) {
                //------------------------------------------------------------------------------------------
                //This statement handles the special case of the first element within the first row
                //------------------------------------------------------------------------------------------
                top = intArray[i-1][j];
                left = intArray[i][j-1];
                if(lab1::checkValueTwo(current, top, left)){
                    intArray[i][j] = intArray[i][j];

                }
                else{
                    intArray[i][j] = static_cast<int>(floor(((top+left)/2.0)+0.5));
                }
            }
            std::cout<< intArray[i][j]<< ' ';


        }

        std::cout << " | " <<std::endl;
    }

    std::cout << spaces <<" | " <<std::endl;


    //------------------------------------------------------------------------------------------
    // The following loop outputs the dark characters for the corrected intArray
    //------------------------------------------------------------------------------------------
    for (int i = 0; i < size; ++i){
        for (int j = 0; j < size; ++j){
            int value = intArray[i][j];
            std::cout << lab1::swapNumforDarkChar(value) << ' ';
        }
        std::cout << " | " <<std::endl;
    }
    std::cout << dashes << "| " <<std::endl;
    std::cout <<std::endl<<std::endl;

}
 bool  lab1::checkValueTwo(int current, int val1, int val2){
    //------------------------------------------------------------------------------------------
    //This method compares a case in which there are only two bordering values to a current value
    //if the difference between the current value and the other two values is greater than one,
    //false is returned to the calling code.
    //
    //------------------------------------------------------------------------------------------

    bool result = true;

    int difference1 = std::abs(current-val1);
    int difference2 = std::abs(current-val2);

    if (difference1 > 1||difference2 > 1) {
        result = false;
        return result;
    }else return result;
}
```

```
bool lab1::checkValueThree(int current, int val1, int val2, int val3){
    //------------------------------------------------------------------------------------
    //This method compares a case in which there are three bordering values to a current value.
    //If the difference between the current value and any of the other three values is greater
    //than one, false is returned to the calling code.
    //------------------------------------------------------------------------------------

    bool result = true;

    int difference1 = abs(current-val1);
    int difference2 = abs(current-val2);
    int difference3 = abs(current-val3);

    if (difference1 > 1){
        return result = false;
    }
    else if (difference2 > 1) {
        return result = false;
    } else if (difference3 > 1) {
        return result = false;
    }else

        return result;
}

bool lab1::checkValueFour(int current, int val1, int val2, int val3, int val4){
    //------------------------------------------------------------------------------------
    //This method compares a case in which there are four bordering values to a current value.
    //If the difference between the current value and any of the other four values is greater
    //than one, false is returned to the calling code.
    //
    //------------------------------------------------------------------------------------

    bool result = true;

    int difference1 = std::abs(current-val1);
    int difference2 = std::abs(current-val2);
    int difference3 = std::abs(current-val3);
    int difference4 = std::abs(current-val4);

    if (difference1>1||difference2>1||difference3>1||difference4>1) {

        return result = false;
    }else

        return result;
}

string lab1::swapNumforDarkChar(int array){
    //------------------------------------------------------------------------------------
    //This method compares the integer input from the calling code, and returns a string.
    //The string returned is the corresponding "Dark Character" representation of the input value.
    //------------------------------------------------------------------------------------

    string result = " ";
    if(array== 0){
        result = " ";
    }
    else if(array== 1){
        result = ".";
    }
    else if(array== 2){
        result = ",";
    }
    else if(array== 3){
        result = "-";
    }
    else if(array== 4){
        result = "!";
    }
    else if(array== 5){
        result = "+";
    }
    else if(array== 6){
        result = "*";
    }
    else if(array== 7){
        result = "#";
    }
    else if(array== 8){
        result = "$";
    }
    else if(array== 9){
        result = "&";
    }
    return result;
}
```

```
:
printf \\n\\n
```

```
cat -b lab1test.cpp
     1	//
     2	//  lab1test.cpp
     3	//  prog1
     4	//
     5	//  Created by James Francis II on 9/9/14.
     6	//  Copyright (c) 2014 James Francis II. All rights reserved.
     7	//

     8	#include "lab1.h"

     9	int main(int argc, const char * argv[])
    10	{
    11	    lab1();
    12	    return 0;
    13	}
:
g++ lab1.cpp lab1test.cpp -o p1
:
p1

Uncorrected Input
---------------------|
7 6 9 4 5 4 3 2 1 0  |
6 5 5 5 6 5 4 3 2 1  |
6 5 6 6 7 6 8 4 3 2  |
1 5 6 7 7 7 6 5 4 3  |
5 5 6 7 6 7 7 6 5 9  |
5 6 7 6 5 6 6 5 4 3  |
5 6 7 9 5 5 6 5 4 3  |
5 5 6 7 6 6 7 6 5 4  |
5 9 5 6 7 6 5 0 3 2  |
5 5 5 5 6 5 4 3 2 7  |
                     |
# * & ! + ! - , .    |
* + + + * + ! - , .  |
* + * * # * $ ! - ,  |
. + * # # # * + ! -  |
+ + * # * # # * + &  |
+ * # * + * * + ! -  |
+ * # & + + * + ! -  |
+ + * # * * # * + !  |
+ & + * # * +   - ,  |
+ + + + * + ! - , #  |
---------------------|
Corrected Input
---------------------|
7 7 5 4 5 4 3 2 1 0  |
6 6 5 5 6 5 5 3 2 1  |
4 5 6 6 7 7 6 4 3 2  |
5 5 6 7 7 7 6 5 4 5  |
5 5 6 7 6 7 7 6 6 5  |
5 6 7 7 6 6 6 5 5 4  |
5 6 7 7 6 5 6 5 4 3  |
5 7 6 7 6 6 6 4 4 3  |
6 6 5 6 7 6 4 4 3 4  |
5 5 5 5 6 5 4 3 4 4  |
                     |
# # + ! + ! - , .    |
* * + + * + + - , .  |
! + * * # # * ! - ,  |
+ + * # # # * + ! +  |
+ + * # * # # * * +  |
+ * # # * * * + + !  |
+ * # # * + * + ! -  |
+ # * # * * * ! ! -  |
* * + * # * ! ! - !  |
+ + + + * + ! - ! !  |
---------------------|
```