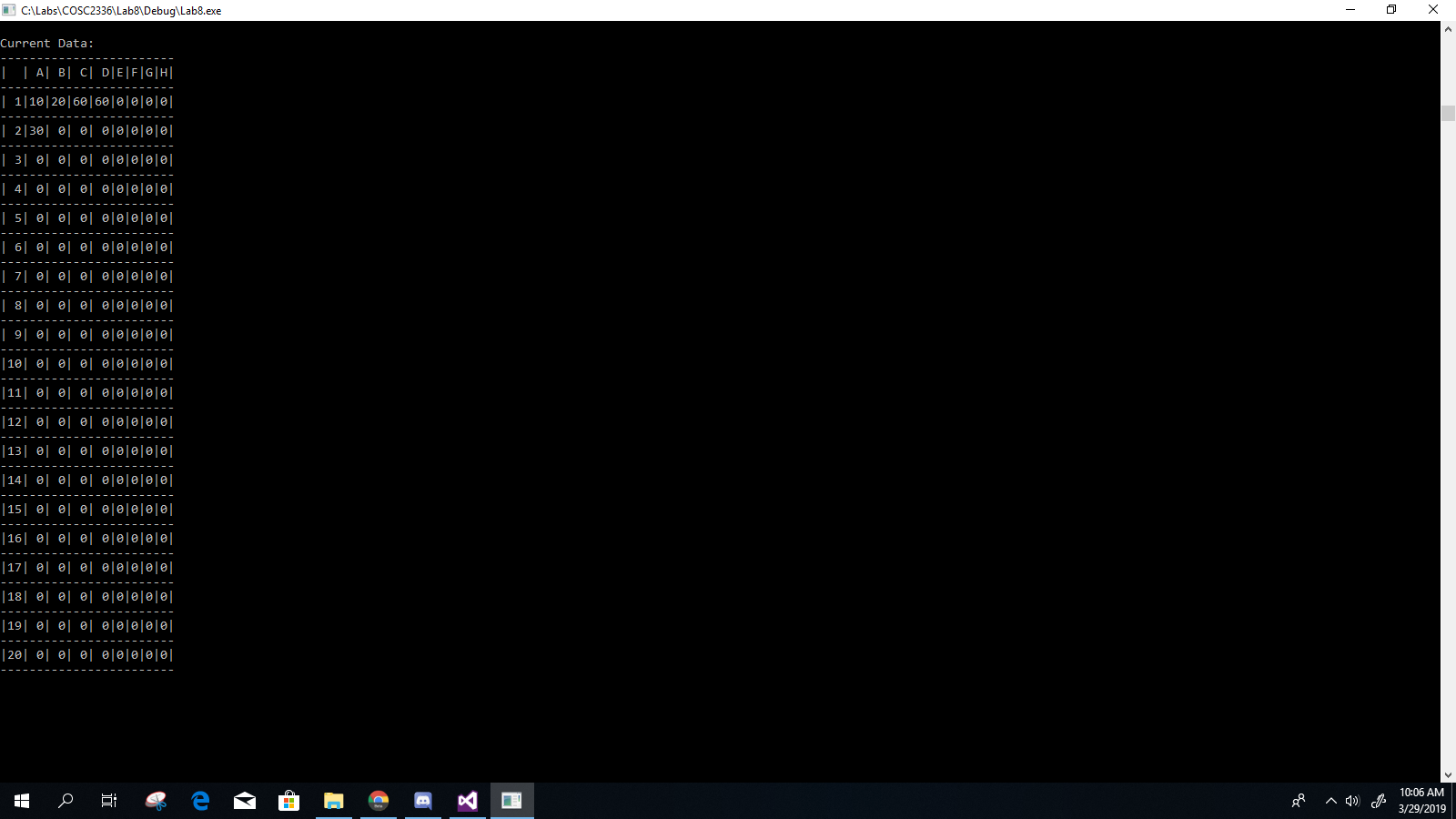
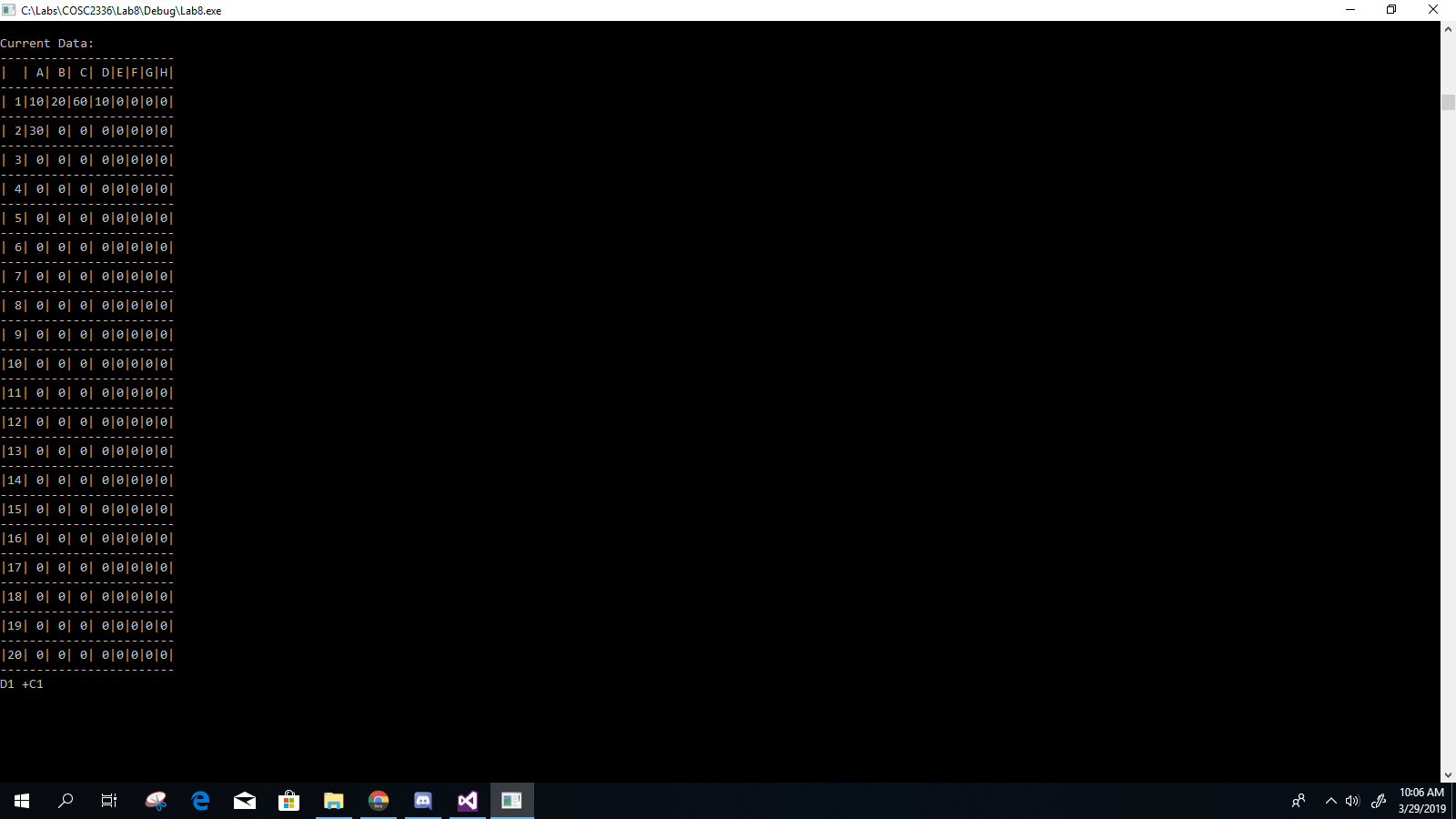
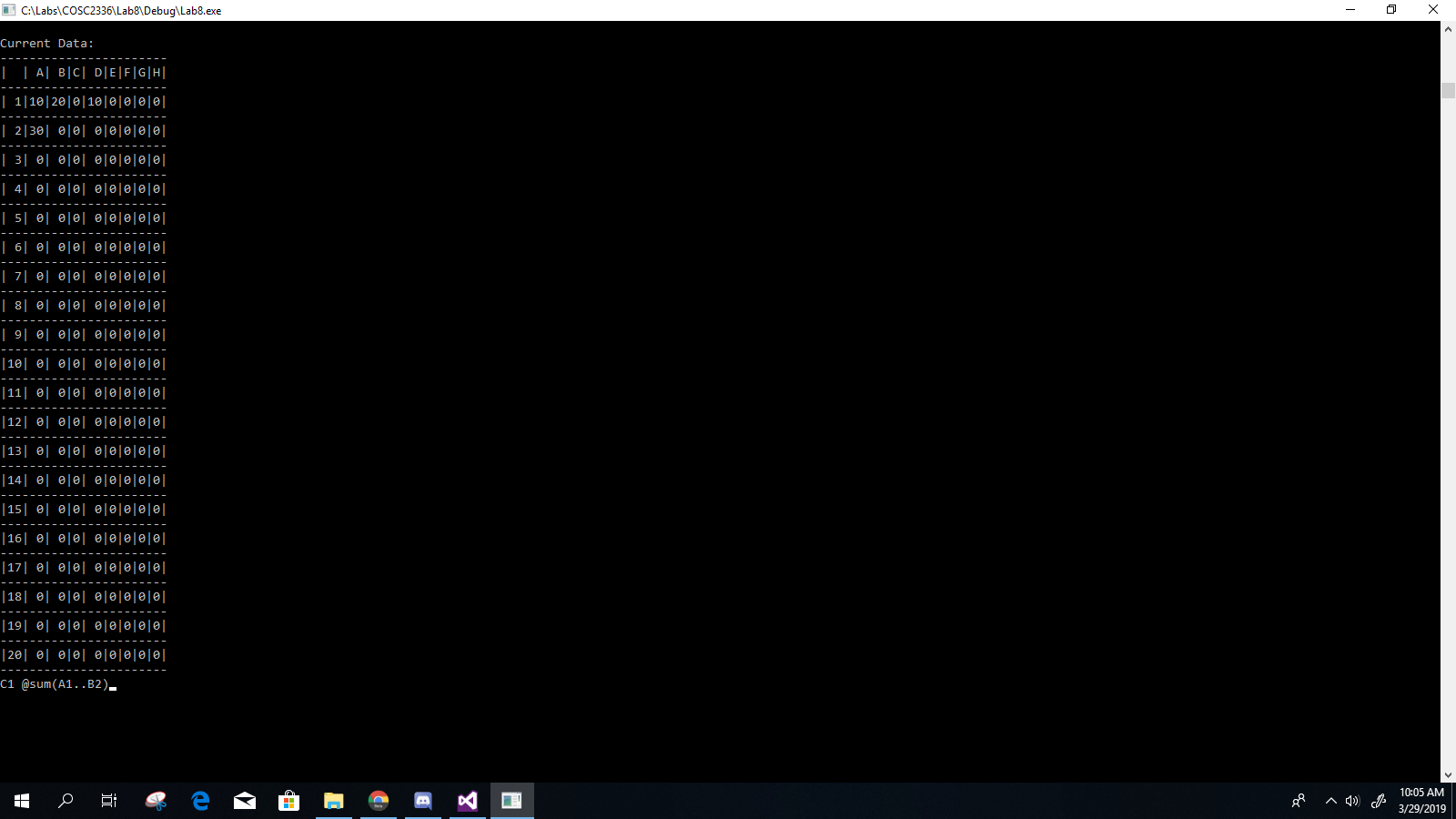
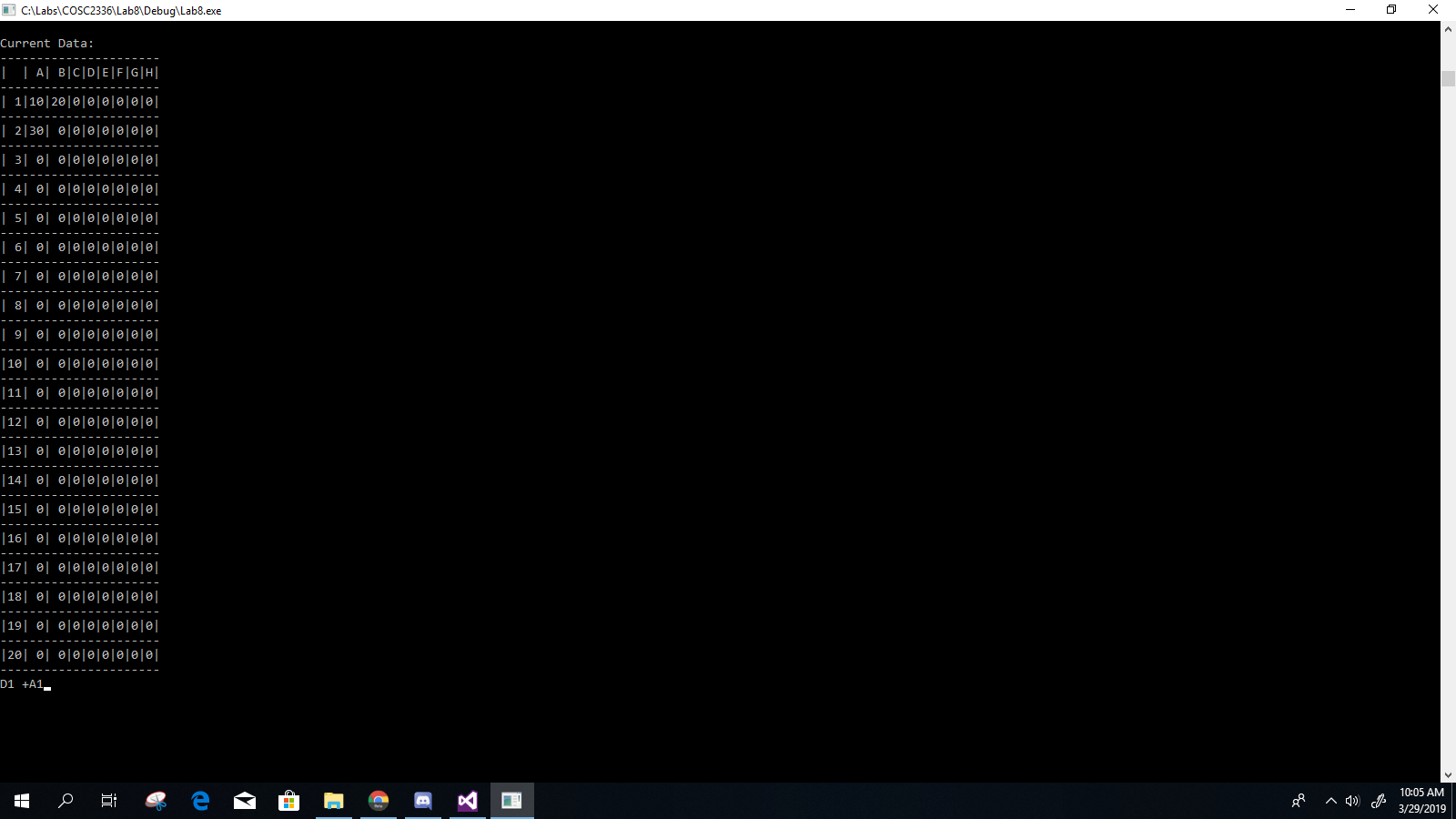
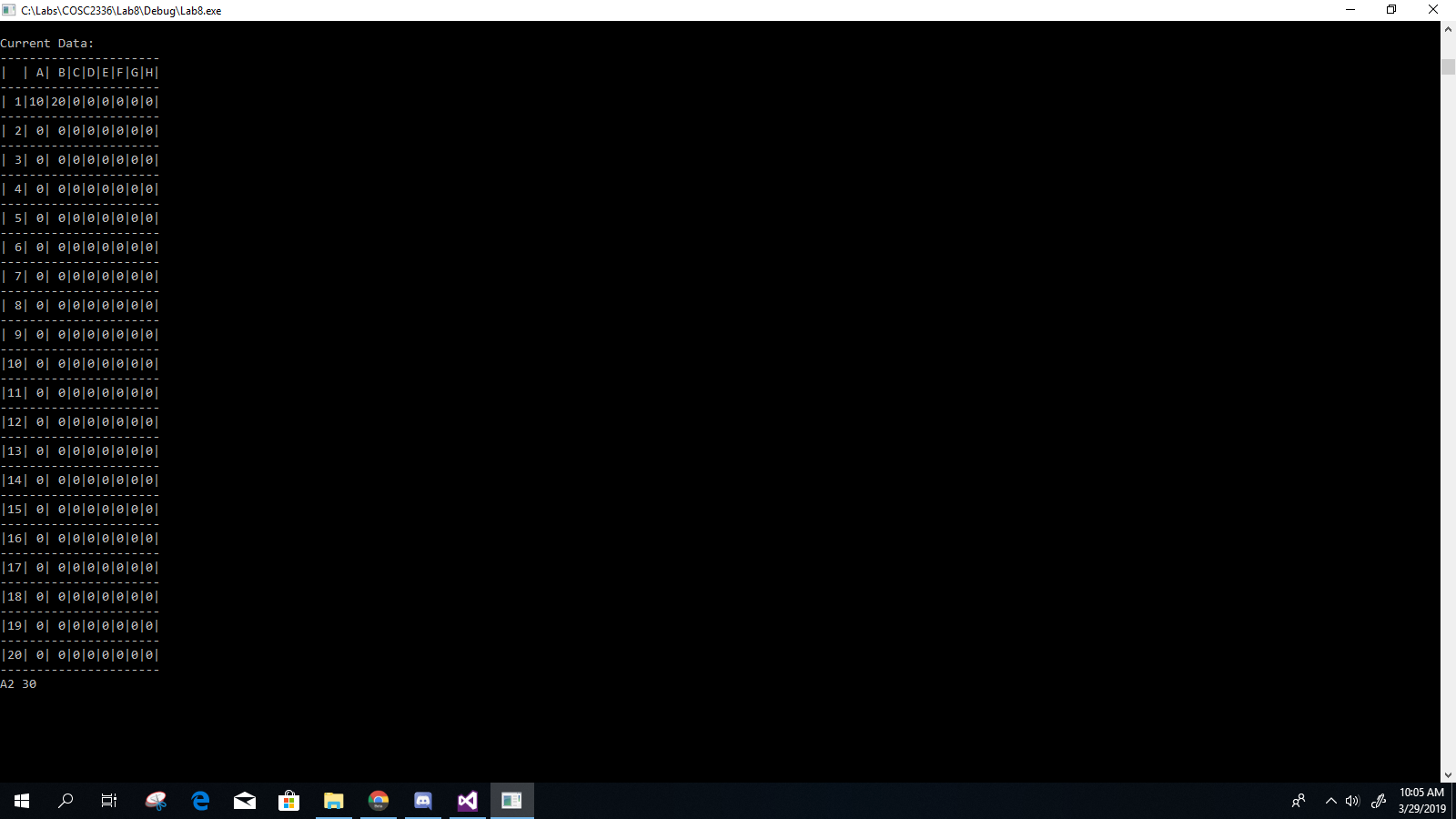
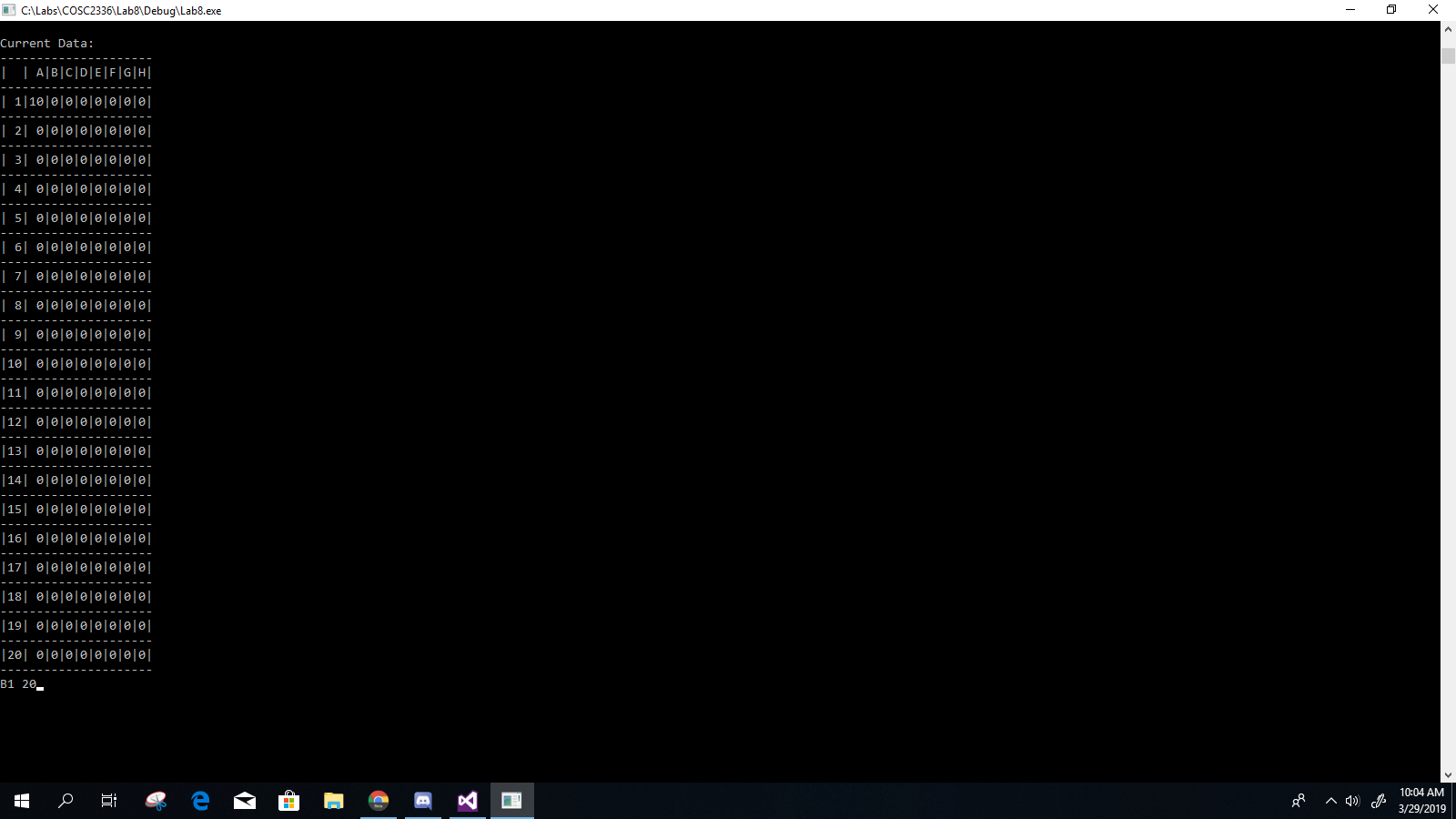
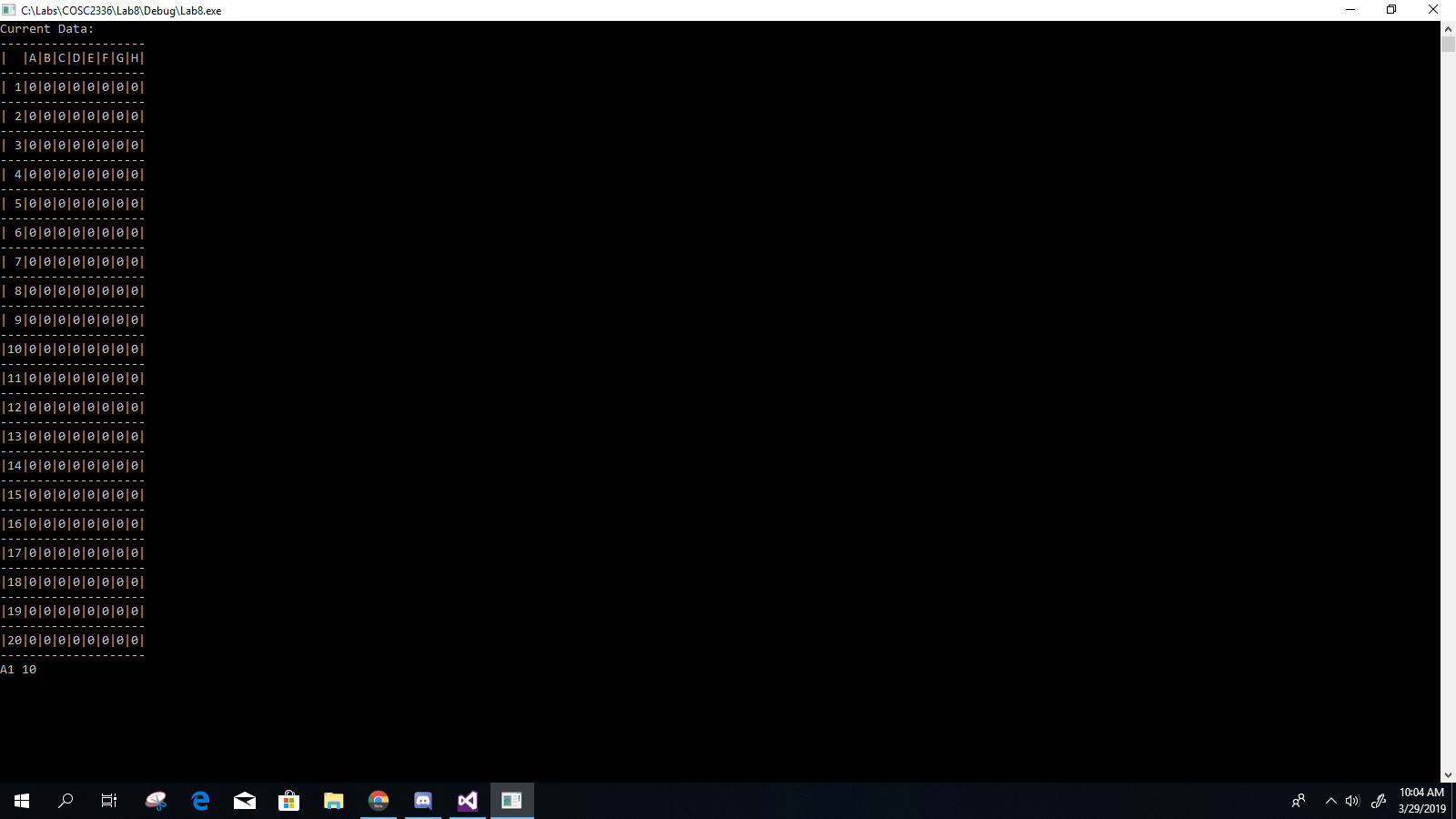
Name: William A. Brannon

Assignment: Lab Assignment Report #8

Date Due: Monday, April, 01, 2019

Class: Fundamentals of Programming III Section #1

**Program Output**



**Lab8.cpp**

// Lab8.cpp : spreadsheets/graphs

// By William A. Brannon on 03/26/2019

#include "stdafx.h"

#include <iostream>

#include <iomanip>

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

class Cell {

public:

int row;

char col;

string val;

Cell() {

}

Cell(int r, char c, string v) {

row = r;

col = c;

val = v;

}

bool operator==(const Cell &r) {

return (row == r.row && col == r.col && val == r.val);

}

bool operator <(const Cell& r) const

{

if (col < r.col) {

return true;

}

else if (col == r.col) {

if (row < r.row) {

return true;

}

else {

return false;

}

}

else if (col > r.col) {

return false;

}

return false;

}

};

class Spreadsheet{

private:

vector<Cell> cells;

public:

Spreadsheet(int r, char c, string v) {

//spreadsheet needs to be initialized with the cell at the bottom-right corner

Cell temp;

temp.row = r;

temp.col = c;

temp.val = v;

cells.push\_back(temp);

};

Cell findCell(int r, char c) {

for (Cell temp : cells) {

if (temp.row == r && temp.col == c) {

return temp;

}

}

Cell temp(r,c,"0");

cells.push\_back(temp);

return temp;

}

vector<Cell> Column(char c) {

vector<Cell> tempVec;

for (Cell temp : cells) {

if (temp.col == c) {

tempVec.push\_back(temp);

}

}

return tempVec;

}

vector<Cell> Row(int r) {

vector<Cell> tempVec;

for (Cell temp : cells) {

if (temp.row == r) {

tempVec.push\_back(temp);

}

}

return tempVec;

}

string valOf(int r, char c) {

string temp = findCell(r,c).val;

if (temp.at(0) == '+') {

temp = temp.substr(1);

char cTemp = temp.at(0);

temp = temp.substr(1);

int rTemp = stoi(temp);

return valOf(rTemp, cTemp);

}

else if (temp.at(0) == '@') {

temp.pop\_back();

vector<int> values = rangeToVals(temp.substr(5));

if (temp.find("max") != temp.npos) {

int k = 0;

for (int v : values) {

if (v > k) {

k = v;

}

}

return to\_string(k);

} else if (temp.find("min") != temp.npos) {

int k = INT\_MAX;

for (int v : values) {

if (v < k) {

k = v;

}

}

return to\_string(k);

} else if (temp.find("avg") != temp.npos) {

int k = 0;

for (int v : values) {

k += v;

}

k /= values.size();

return to\_string(k);

} else if (temp.find("sum") != temp.npos) {

int k = 0;

for (int v : values) {

k += v;

}

return to\_string(k);

}

}

else {

return temp;

}

return temp;

};

vector<int> rangeToVals(string s) {

//valid range format is top-left cell and bottom right cell (inclusive) ex: "A1..B2"

string delimiter = "..";

size\_t pos = s.find(delimiter);

string s1 = s.substr(0, pos);

s.erase(0, pos + delimiter.length());

string s2 = s;

char c1 = s1.at(0);

char c2 = s2.at(0);

s1 = s1.substr(1);

s2 = s2.substr(1);

int r1 = stoi(s1);

int r2 = stoi(s2);

vector<Cell> columns;

vector<Cell> rows;

for (char c = c1; c <= c2; c++) {

vector<Cell> temp = Column(c);

for (Cell cell : temp) {

columns.push\_back(cell);

}

}

for (int r = r1; r <= r2; r++) {

vector<Cell> temp = Row(r);

for (Cell cell : temp) {

rows.push\_back(cell);

}

}

vector<int> output;

for (Cell c : columns) {

for (Cell r : rows) {

if (c == r) {

output.push\_back(stoi(valOf(r.row, r.col)));

}

}

}

return output;

};

void input(string s) {

string delimiter = " ";

size\_t pos = s.find(delimiter);

string s1 = s.substr(0, pos);

s.erase(0, pos + delimiter.length());

string s2 = s;

char c = s1.at(0);

s1 = s1.substr(1);

int i = stoi(s1);

int j = 0;

Sort();

Cell target = findCell(i, c);

for (Cell temp : cells) {

if ((temp.row == i) && (temp.col == c)) {

cells.at(j).val = s2;

}

j++;

}

}

void Sort() {

std::sort(cells.begin(), cells.end());

}

void Display() {

Sort();

Cell size = sizeOf();

cout << "Current Data: " << endl;

horizontalLine(size);

header(size);

for (int i = 1; i <= size.row; i++) {

horizontalLine(size);

printRow(i, size);

}

horizontalLine(size);

}

void horizontalLine(Cell size) {

for (char c = '@'; c <= size.col; c++) {

cout << "-";

int i = colWidth(c);

for (int j = 0; j < i; j++) {

cout << "-";

}

}

cout << "--" << endl;

}

void header(Cell size) {

cout << "|" << setw(to\_string(sizeOf().row).length()) << "" << setw(1);

for (char c = 'A'; c <= size.col; c++) {

cout << "|" << setw(colWidth(c)) << c << setw(1);

}

cout << "|" << endl;

}

void printRow(int r, Cell size) {

cout << "|" << setw(to\_string(sizeOf().row).length()) << r << setw(1);

for (char c = 'A'; c <= size.col; c++) {

cout << "|" << setw(colWidth(c)) << valOf(r,c) << setw(1);

}

cout << "|" << endl;

}

void consoleInput() {

string temp;

getline(cin, temp);

input(temp);

}

int colWidth(char c1) {

unsigned int temp = 1;

vector<Cell> column = Column(c1);

for (Cell c : column) {

if (valOf(c.row,c.col).length() > temp) {

temp = valOf(c.row, c.col).length();

}

}

return temp;

}

Cell sizeOf() {

Cell temp(1,'A',"");

for (Cell c : cells) {

if (c.row > temp.row) {

temp.row = c.row;

}

if (c.col > temp.col) {

temp.col = c.col;

}

}

return temp;

}

};

void clearConsole() {

for (int i = 0; i < 100; i++) {

cout << endl;

}

}

int main()

{

Spreadsheet tester(20,'H',"0"); //initialization of size (only necessary because of display)

while (true) {

tester.Display();

tester.consoleInput();

clearConsole();

};

//system("pause");

return 0;

}