```
// Spencer Harper
// Output data based off the crime statistics of Baltimore

#include <iostream>
#include <iomanip>
#include <string>
#include <fstream>
#include <sstream>
#include <vector>

using namespace std;

void PrintLine(string, int, int, int);
void PrintHeader(string, string, string, string);
string FormatString(string);
int SearchDistrict(string district_name, const vector<string> &);
string min(vector<int> &, vector<int> &);
string max(vector<int> &, vector<int> &);
void PoliticalSummary(vector<string> &, vector<int> &, vector<int> &, vector<int> &, vector<int> &);

int main(int argc, char *argv[]) {

    ifstream fin;
    istringstream sin;
    string line, district_name, response;
    double nonv_crime, v_crime, d_crime;
    int i, num_lines=0, size, index;
    vector<string> district_names;
    vector<int> nonviolent_crimes, violent_crimes, domestic_crimes, indexes;

    // check to see if user entered the correct command line arguments
    if(argc != 3) {
        cout << "Invaid command line arguments";
        return -1;
    }

    // convert command line argument into an integer
    sin.str(argv[2]);
    sin >> size;
    sin.clear();

    // open and check file
    fin.open(argv[1]);
    if(fin.fail()) {
        cout << "file failed to open";
        return -1;
    }

    // add data entries to vectors
    while(getline(fin, line)) {

        num_lines++;
        sin.str(line);

        sin >> district_name >> nonv_crime >> v_crime >> d_crime;

        district_names.push_back(district_name);
        nonviolent_crimes.push_back(nonv_crime * 10);
        violent_crimes.push_back(v_crime * 10);
        domestic_crimes.push_back(d_crime * 10);

        sin.clear();

    }
```

```
    fin.close();

    // if user prompts a number larger than file size print full file
    if(size > district_names.size())
        size = district_names.size();

    // print header information
    cout << "Top " << size << " Crime Ridden Areas In Baltimore" << endl;
    PrintHeader("District", "NVC", "VC", "DV");
    cout << endl;

    // print the number of districts the user entered at the command line
    for(i=0; i<size; i++) {
        PrintLine(district_names[i], nonviolent_crimes[i], violent_crimes[i], domestic_crimes[i]);
    }

    // prompt the user to enter distrits to get stats for that district
    // if district does not exist print '----'
    // continue asking until the user enters 'done'
    cout << endl << "Enter a district: ";
    cin >> response;
    response = FormatString(response);
    while(response != "Done") {
        index = SearchDistrict(response, district_names);
        PrintHeader("", "NVC", "VC", "DV");
        if(index != -1) {
            PrintLine(response, nonviolent_crimes[index], violent_crimes[index],
domestic_crimes[index]);
            indexes.push_back(index);
        }
        else
            PrintHeader(response, "----", "----", "----");

        cout << endl << "Enter a district: ";
        cin >> response;
        response = FormatString(response);

    }

    // print political summary of previous user requests
    PoliticalSummary(district_names, nonviolent_crimes, violent_crimes, domestic_crimes, indexes);

}

// prints formatted crime statistics
void PrintLine(string district, int nonviolent, int violent, int domestic) {

    cout << left << setw(26) << district;
    cout << left << setw(5) << violent;
    cout << left << setw(5) << domestic;
    cout << left << setw(5) << nonviolent << endl;

}

// same functionality as PrintLine but this function will accept strings instead of ints
void PrintHeader(string district, string nonviolent, string violent, string domestic) {

    cout << left << setw(26) << district;
    cout << left << setw(5) << violent;
    cout << left << setw(5) << domestic;
    cout << left << setw(5) << nonviolent << endl;

}
```

```
// make every letter lowercase except the first
string FormatString(string str) {

    int i, size;
    str[0] = toupper(str[0]);
    size = str.length();

    for(i=1; i<size; i++) {
        str[i] = tolower(str[i]);
    }

    return str;

}


// search for index of district name while ignoring case
int SearchDistrict(string district_name, const vector<string> & dist_vec) {

    int i, index=-1, size;

    size = dist_vec.size();
    district_name = FormatString(district_name);

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

        if(district_name == FormatString(dist_vec[i])) {
            index = i;
            break;
        }

    }

    return index;

}


// find the minimum crime rate based off of given indexes
string min(vector<string> & names, vector<int> & crimes, vector<int> & indexes) {
    int i, min, mindex, size=indexes.size();

    mindex = indexes[0];
    min = crimes[mindex];
    for(i=1; i<size; i++) {
        if(min > crimes[indexes[i]]) {
            mindex = indexes[i];
            min = crimes[mindex];
        }
    }

    return names[mindex];

}

// find the maximum crime rate based off of given indexes
string max(vector<string> & names, vector<int> & crimes, vector<int> & indexes) {
    int i, max, mindex, size=indexes.size();

    mindex = indexes[0];
    max = crimes[mindex];
    for(i=1; i<size; i++) {
        if(max < crimes[indexes[i]]) {
            mindex = indexes[i];
            max = crimes[mindex];
```

```
        }
    }

    return names[mindex];

}

// print the highest and lowest crime rates for each type of crime using the min and max functions
void PoliticalSummary(vector<string> & d_names, vector<int> & nonv_crimes, vector<int> & v_crimes,
vector<int> & dom_crimes, vector<int> & indexes) {

    cout << endl;
    cout << "Highest Nonviolent Crime: " << max(d_names, nonv_crimes, indexes) << endl;
    cout << "Lowest Nonviolent Crime : " << min(d_names, nonv_crimes, indexes) << endl << endl;

    cout << "Highest Violent Crime: " << max(d_names, v_crimes, indexes) << endl;
    cout << "Lowest Violent Crime : " << min(d_names, v_crimes, indexes) << endl << endl;

    cout << "Highest Domestic Crime: " << max(d_names, dom_crimes, indexes) << endl;
    cout << "Lowest Domestic Crime : " << min(d_names, dom_crimes, indexes) << endl << endl;

}
```