Homework #3

Complete By: Monday, September 21st @ 11:59pm
Assignment: completion of following exercise
Policy: Individual work only, late work *is* accepted
Submission: electronic submission via Blackboard

Assignment

We are going to analyze Chicago crime data. There are 3 text files in CSV format (comma-separated values). The first, "Crimes.csv", contains crime data, one crime per row:

```
Date,IUCR,Arrest,Domestic,Beat,District,Ward,Community,Year
09/03/2015 11:57:00 PM,0820,true,false,0835,008,18,66,2015
09/03/2015 11:55:00 PM,1512,true,false,0931,009,20,61,2015
09/03/2015 11:18:00 PM,141A,false,false,1423,014,26,24,2015
.
.
.
```

This file is 300MB in size, so a smaller file is also provided for easier testing: "Crimes-2014-2015.csv". These files contain real data, which means don't expect perfectly "clean" data. For example, you can expect some columns to be missing:

```
01/22/2001 09:00:00 PM,2110,true,false,1822,018,,,2001
```

Your program should not crash if a data field is missing (the comma will be there, it's just that the column may be empty).

The second file, "IUCR-codes.csv", contains the definition of the IUCR codes — Illinois Uniform Crime Reporting codes. Here's the format:

```
IUCR,PRIMARY_DESCRIPTION,SECONDARY_DESCRIPTION
0110,HOMICIDE,FIRST DEGREE MURDER
0130,HOMICIDE,SECOND DEGREE MURDER
.
.
031A,ROBBERY,ARMED: HANDGUN
.
.
```

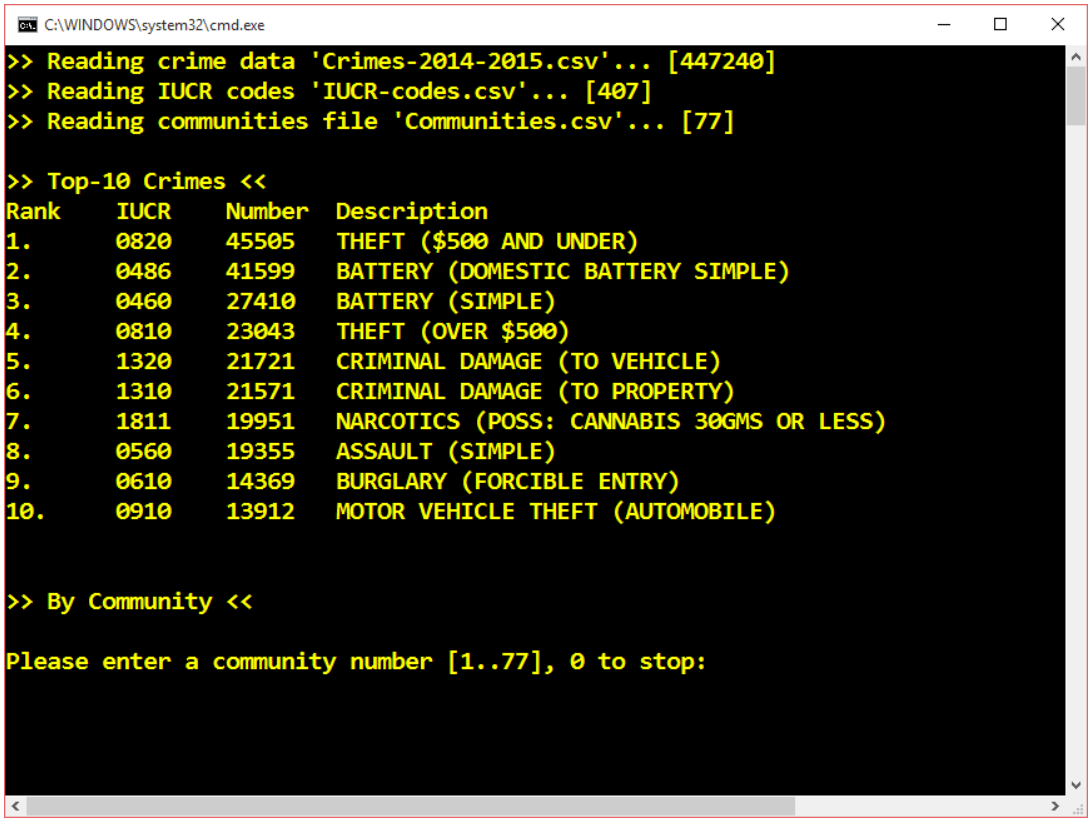
Note that an IUCR code is not an integer, since some of the codes contain letters (e.g. "031A"). Also, as you might expect, the Crime data and IUCR codes are not in sync. In particular, there are crimes with IUCR codes, e.g. 5114, that are not listed in "IUCR-codes.csv". This should not cause your program to crash.

The third and final file, "Communities.csv", contains the number and name of each community in Chicago. There are 77:

```
Number,Community
1,Rogers Park
2,West Ridge
3,Uptown
.
.
.
76,Ohare
77,Edgewater
```

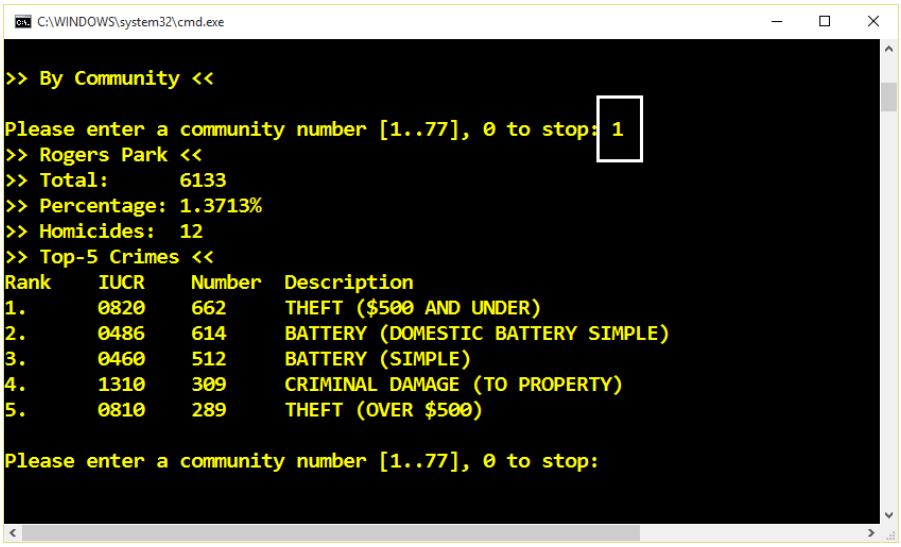
Just in case you're curious, there's an associated PDF file ("Communities.pdf") to show you the location of each community in Chicagoland.

The assignment is to analyze this data in three different ways. Part 1 is to input the data and compute the top-10 crimes throughout the city of Chicago. Here are the results for the smaller "Crimes-2014-2015.csv" file:

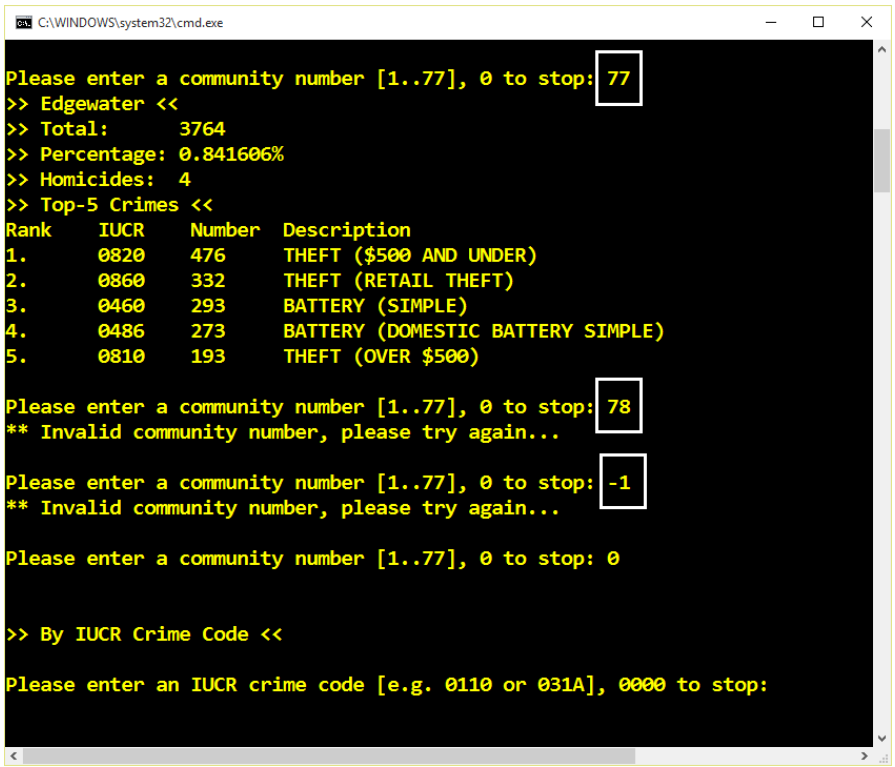


Your output should match this output as close as possible; we reserve the right to deduct points if your output is wrong, or differs from the format shown above (which makes it harder for our TAs to grade). The columns are created by output "\t" characters; the Description column is a combination of the Primary and Secondary

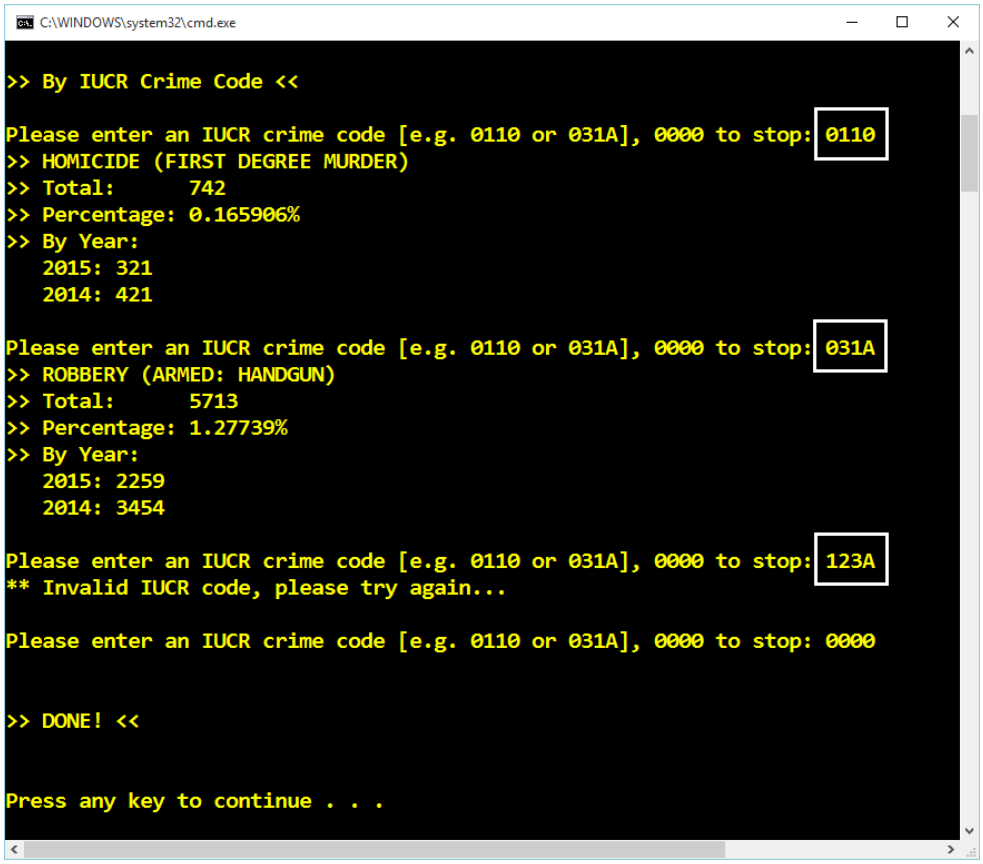
descriptions from the "IUCR-codes.csv" file. Part 2 is to allow the user to analyze the data by community. Here are some examples (continuing from the previous screenshot):



And another:



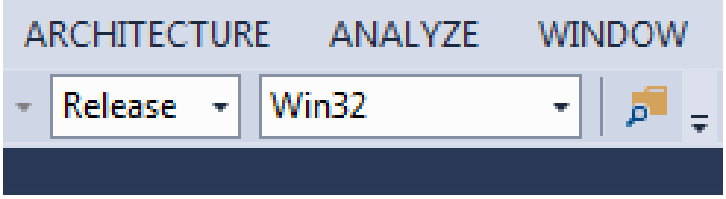
For a given community, output the total # of crimes that occurred in that community, the percentage this represents of the overall crime in Chicago, the total # of homicides (for which there are 4 IUCR codes: 0110, 0130, 0141, and 0142), and then the top-5 crimes for that community. Finally, part 3 is to allow the user to analyze the data by crime, denoted by entering an IUCR code:



For a valid IUCR code, output the primary and secondary description, the total # of occurrences of that crime, the percentage that represents in overall Chicago crime, and a yearly breakdown (total per year). Assume that the "Crimes.csv" file we test against will be in descending order by year, e.g. 2015, 2014, …, 2001. But do not assume any particular sequences of years, we may test against a file containing data 1910, 1909, …, 1890.

Note that we are providing text files for you to test against; see "hw3-data-files.zip" on the course web page (under "Homeworks"). However, we reserve the right to test against different text files to ensure you have written a general-purpose program. Two different crimes files are provided the larger 300+ MB file is provided so you can test the efficiency of your solution. While you are not required to provide a sub-second solution for the 300+ MB file, you are expected to (a) run your program against this large input file, and (b) improve your program as necessary so execution time < 5 minutes. Once again it's not just about solving the problem, but solving it reasonably well.

For best execution times in Visual Studio, change the "Solution Configuration" drop-down in the toolbar from "Debug" to "Release", and then use Ctrl+F5 to run ("Start without debugging").



Assignment Details

Once again, it's not about just solving the problem, but how you solve it. The goal is to use the modern features of C++, not revert back to C-style programming. In particular, you must satisfy *all* of the following requirements:

1. Use range-based for loops, or iterator-based loops; resort to index-based loops only when necessary (e.g. an index-based loop makes sense when you output the top-10 crimes).
2. Do not open/close the file yourself. Instead, use an ifstream approach as discussed in lecture, or the approach shown in the next section.
3. Use classes to store the data.
4. Use the built-in C++ containers to store your objects, e.g. vector<T> is always a good choice. However, feel free to use any of the built-in containers.
5. Use the built-in C++ algorithms. When you need to sort — e.g. to compute the top-10 — use std::sort. Writing your own sort algorithm will be a -15 point deduction. Need to count how many crimes occurred in a community? Use std::count_if. Need to search? Use std::find, std::find_if, or std::binary_search. Writing your own count and search functions will be a -15 point deduction.
6. Write and call functions; you need to define at least ***three*** (3) functions that take one or more of your containers as parameters. Think about the parameter passing.

Avoid the use of **new** or **malloc**, that's a symptom of C-style thinking.

If you haven't already, I highly recommend using Visual Studio for this assignment. For help, see the "*Getting Started with Visual Studio*" document on the course web page (under "Misc"). That said, you can solve this assignment without using Visual Studio since it's based on standard C++11 or C++14. Note that you cannot use http://ideone.com, since it doesn't support input files.

Some Helpful C++

For simplicity, you can define all your classes, functions, and main() program in a single "main.cpp" file. However, feel free to be more modular if you wish, but it's not required.

The input files in this assignment contain spaces — e.g. spaces within a community name or crime description. This implies that if you read the file using a string-based iterator, you will read word by word, not line by line. To make life easier, read the file line by line using ***std::getline*** (you will need to #include <string>):

```
ifstream  ifstreamFile("filename");
string    line;

while ( std::getline(ifstreamFile, line) )  // read file line by line:
{
   // do something with line
}
```

Parsing input files is always troublesome. Here's a more modern approach based on C++'s *string stream*. The example below defines a simple **Review** class, along with a **ParseReview** function that given a line of data in CSV format, parses that line and returns a Review object.

```
#include <iostream>
#include <string>
#include <sstream>

using namespace std;

//
// Stores one review:
//
class Review
{
public:
  int movieID;
  int userID;
  int movieRating;

  Review(int movie, int user, int rating)
  {
    movieID = movie;
    userID = user;
    movieRating = rating;
  }
};

//
// Parses one line in CSV format and returns a Review object:
//
Review ParseReview(string line)
{
  istringstream strstream(line);
  string movieIDstr, userIDstr, ratingstr;

  std::getline(strstream, movieIDstr, ',');
  std::getline(strstream, userIDstr, ',');
  std::getline(strstream, ratingstr, ',');

  int movieID = std::stoi( movieIDstr );
  int userID = std::stoi( userIDstr );
  int movieRating = std::stoi( ratingstr );

  return Review(movieID, userID, movieRating);
}
```

Don't forget that some of the data in a line could be missing, you'll need to figure out how to account for that…

Electronic Submission

Using Blackboard (http://uic.blackboard.com/), submit your solution under the assignment "HW3". If you have a single file just attach that file, otherwise create an archive file (.zip) of your program folder and submit the archive. The top of your main source file should contain a header comment like the following:

```
//
// C++ program to analyze Chicago crime data.
//
// <<YOUR NAME HERE>>
// U. of Illinois, Chicago
// CS341, Fall 2015
// Homework 3
//
```

Your program should be readable with proper indentation and naming conventions; commenting is expected where appropriate. You may submit as many times as you want before the due date, but we grade the last version submitted. This implies that if you submit a version before the due date and then another after the due date, we will grade the version submitted after the due date — we will *not* grade both and then give you the better grade. We grade the last one submitted. In general, do not submit after the due date unless you had a non-working program before the due date.

Have a question? Use Piazza, not email

As discussed in the syllabus, questions must be posted to our course Piazza site — questions via email will be ignored. Remember the guidelines for using Piazza:

1. *Look before you post* — the main advantage of Piazza is that common questions are already answered, so search for an existing answer before you post. Posts are categorized to help you search, e.g. "Pre-class" or "HW".
2. Post publicly — only post privately when asked to by the staff, or when it's absolutely necessary (e.g. the question is of a personal nature). Private posts defeat the purpose of piazza, which is answering questions to the benefit of everyone.
3. Ask pointed questions — do not post a big chunk of code and then ask "help, please fix this". Staff and other students are willing to help, but we aren't going to type in that chunk of code to find the error. You need to narrow down the problem, and ask a pointed question, e.g. "on the 3rd line I get this error, I don't understand what that means…".
4. Post a screenshot — sometimes a picture captures the essence of your question better than text. Piazza allows the posting of images, so don't hesitate to take a screenshot and post; see http://www.take-a-screenshot.org/ .
5. Don't post your entire answer — you just gave away the answer to the ENTIRE CLASS. Sometimes you will need to post code when asking a question, but then post only the fragment that denotes your question, and omit whatever details you can. If we can't answer your question, we will ask you to post your entire code privately — i.e. "visible to staff only". This is an option when you post.

Policy

Late work *is* accepted. You may submit as late as 24 hours after the deadline for a penalty of 25%. After 24 hours, no submissions will be accepted.

All work is to be done individually — group work is not allowed. While I encourage you to talk to your peers and learn from them (e.g. your "iClicker teammates" or Piazza), this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is described here:

http://www.uic.edu/depts/dos/docs/Student%20Disciplinary%20Policy.pdf .

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone else's iClicker with the intent of answering for that student, whether for a quiz, exam, or class participation. Academic dishonesty is unacceptable, and penalties range from failure to expulsion from the university; cases are handled via the official student conduct process described at http://www.uic.edu/depts/dos/studentconductprocess.shtml .