ECS 36C Winter 2019
Due: Jan 23, 2019

Program 1

Objectives of the assignment

The objectives of this programming assignment are:

- Reviewing most of the concepts learned in ECS 30 and ECS 40 (object-oriented programming, file manipulation, command line arguments, operator overloading, Makefile, etc.)
- Understanding how computational complexity can apply to real-life programs.
- Running test measurements and analysing the results in order to provide an argued answer to the given problem.
- Writing high-quality C++ code by following established industry standards.

Program description

Introduction

You are helping the library catalog books, and provide books to departments upon request..

The librarian received new shipments of books with different ISBN weekly, they also receive requests for new books with different ISBN numbers in either new, used, or digital format. (ISBN: this is the number the publisher assigns An ISBN is assigned to each edition and variation of a book)

For example the Library may receive a shipment of 5 books

$ cat newbooks.dat

55, new

66, used

67, digital

01, new

12, digital

$cat request.dat

67, digital

67, new

ECS 36C Winter 2019
Due: Jan 23, 2019

Programing Assignment:

Create a program SearchNewBooks.cc that given as command line input newbooks.dat and request.dat and a method of search either binary or linear, creates a file of the number of books found from the request list

```
$SearchNewBooks  newbooks.dat request.dat  //run program
$Choice of search method ([l]inear, [b]inary)?
1
CPU time: 10.777 microseconds

$cat found.dat
1
```

Written:

By running the program multiple times and with inputs of different size explain the pros and cons of linear search versus sorting first and using binary search, with respect to computational complexity. Try to explain if it is possible to determine a ratio between the number of new books and the number of requests to search, that makes either linear search or binary search the best option.

Constraints

Your program must be compatible with C++ only use the standard functions provided by the C++ Standard Library.
Your program cannot be linked to any other external libraries (e.g., Boost).
Your code must follow the Google C++ Style Guide and be properly commented when necessary.

Grading

Your grade for this assignment will be broken down in several scores:

- Grading Guide:
  - 75% of graded by autograder (**autograder will not be provided for this program, but I will post the executable of for reference and you can use the diff command to determine exact matching of results)**
  - 5% Makefile

- 5 % of grade - code style (for example you lose points for not creating functions for reusable code
- 15% written assignment

Suggested work phases

Phase 0: preliminary work

Copy files

In this preliminary phase, copy the few provided files from /home/yafrid/cs36c/public/p1/ to your project directory:

- create_testData: a program that generates fake datasets in order to test your program with inputs of various sizes.
  - It expects two command line arguments when being run:
    - The number of newBooks to generate in the output newbooks.dat file
    - The number of requested books to generate in the output requests.dat file
  - For example:

```
$ create_testData 5 3

$ ls newbooks.dat

newbooks.dat

$ create_testData 5 3

$ cat newbooks.dat

58,new

92,used

82,digital

114,used

26,new

$ cat request.dat
```

8,new

1,new

95,new

- CPPLINT.py -- cpplint, the C++ linter developed by Google to ensure that code conforms to their style guide.
- With this python file located in the same directory as your program, you can run the linter as shown in the example below and fix the reported warnings:
- $ python cpplint.py SearchNewBooks.cc
  Done processing SearchNewBooks.cc
- Total errors found: 0
  $

Makefile

Write a simple Makefile that generates the executable SearchNewBooks from the file SearchNewBooks.cc, using g++.
The compiler should be run with the -Wall (enable all warnings) and -Werror (treat all warnings as errors) options.
There should also be a clean rule that removes any generated files (executables, datasets, etc.) and puts the directory back in its original state.

Phase 1: reading in the dataset

In this phase, your program should receive two filenames as command line arguments, the new books file and request file, which the program should read in. The list of new books and list of requested books should should be stored in only two std::vector containers.
Because some code to open these two files will probably be equivalent, you are encourage to write generic functions in order to avoid code duplication.

New Books file

The first file contains a list new books identified by ISNB number and type, one per line. Each line has an integer values and a string.
Example of new books file:

```
$cat newBooks.dat
58,new
92,used
```

```
82,digital
114,used
26,new
```

You should write a class for representing book as a combination of both the ISNB number and type, so that each vector is an object of that class. For debugging purposes, you can overload the << operator to print positional vectors in a pretty way.

Requested Books file

The second file contains a list of Requested books, one per line and expressed as an integer value.
Example of file:

```
$ cat request.dat
8,new
1,new
95,new
```

Phase 2: searching strategies

In this phase, you will need to implement two search strategies in order to find the given requested books in the list of new arrivals: a **linear search and a binary search**.
The user will choose between using one of these two strategies every time the program is run. If the user input is invalid, they should be asked again, as shown in the example below. The error message should be printed to std::cerr.

```
$ ./SearchNewBooks newBooks.dat request.dat result1.dat
Choice of search method ([l]inear, [b]inary)?
J
Incorrect choice
2
Incorrect choice
l
CPU time: 10.777 microseconds
$
```

The general search algorithm is:

```
count = 0
for (b in request)
   if (search if b is in newBooks)
```

```
    count += 1
```

For the binary search, the vector of new books should be sorted by their ISNB number beforehand. For the sorting, you should use std::sort. If you overload the < operator of your positional vector class, this function can easily be applied for sorting all of objects contained in a vector container:
std::sort(vect_list.begin(), vect_list.end());

If two books ISNB numbers are equal, you can order them by “digital”, “new”, and “used”.

Phase 3: processing time measurements

Create a class for counting the number of elapsed microseconds, which you will use to measure the performance of the search strategies. Your class should a high resolution clock, as defined by std::chrono::high_resolution_clock, and convert the time into microseconds for display. Because the linter provided by Google doesn’t like the use of chrono, we have to turn the linter down for the include line specifically as shown in the example.
Here is an example of how to use the relevant functions defined in std::chrono:

```
#include <chrono> // NOLINT (build/c++11)
...
std::chrono::high_resolution_clock::time_point start;
start = std::chrono::high_resolution_clock::now();
... /* some computation happening */
auto end = std::chrono::high_resolution_clock::now();
double elapsed_us = std::chrono::duration<double, std::micro>(end - start).count();
std::cout << "CPU time: " << elapsed_us << " microseconds";
...
```

If you declare ct as an object of such class, your program’s main fragment of code could look like:

```
print "Choice of search method ([l]inear, [b]inary)?"
get choice from user
ct.Reset()
switch (choice)
   case 'l':
      run linear search
      break;
   case 'b':
      sort vector of books
      run binary search
      break;
   default:
```

```
    print "Incorrect choice"
    go back to getting user choice
print "CPU time: " << ct.CurrentTime() << " ticks"
```

Optionally, you can overload the class' << operator and simply display object ct directly (e.g.: std::cout << ct << std::endl;).

Phase 4: writing the result

Your program should now receive a third and last command line argument, specifying the name of the output file.
The number of requested books found in the list of new arrivals should be written in the output file.

```
$SearchNewBooks  newbooks.dat request.dat  //run program
            $Choice of search method ([l]inear, [b]inary)?
            1
            CPU time: 10.777 microseconds

            $cat found.dat
            1
```

Error management

At this point, your program should handle errors properly.
In case of any errors, i.e., if the wrong number of arguments is given or if a file cannot be opened, your program should print the relevant error message on std::cerr as shown in the example below.

```
$ ./SearchNewBooks
Usage: .SearchNewBooks <newBooks.dat> <request.dat> <result_file.dat>
$ ./SearchNewBooksh toto
Usage: .SearchNewBooks <newBooks.dat> <request.dat> <result_file.dat>
$ ./SearchNewBooks toto titi tata
Error: cannot open file toto
$
```

Phase 5: measurement analysis

In this last phase, you need to generate various datasets of different sizes and run them with your program. It is advised to run each dataset at least three times and average the running time to make sure the measurements are steady.

Find a collection of datasets that allows to make some analysis regarding the choice between the two searching strategies.
Write everything in a PDF document name report.pdf that you will submit via gradescope.

Submission

Since we will use auto-grading scripts in order to test your program, it has to strictly follow the specified output format.

Content

You should submit the following files:

- author.txt: your student ID, last name, first name on one line.
- Makefile: a Makefile that compiles your source code without any errors or warnings (on CSIF computers), and builds an executable named SearchNewBooks.
- SearchNewBooks.cc and any other source code files if applicable. For this project, a unique file should be enough though.

Do not submit any other files to csif. Please submit report to gradescope.

Handin

All of your files have to be submitted via handin from one of the CSIF computers:
$ handin cs36c p1 <files>
...
$

You can verify that the files have been properly submitted:
$ handin cs36c p1
The following input files have been received:
...
$

Academic integrity

You are expected to write this project **from scratch**, thus avoiding to use any existing source code available on the Internet, and **alone**. Asking someone else to write your code (e.g., on website such as Chegg.com) is not acceptable and will result in severe sanctions.

You must specify in your report any sources that you have viewed to help you complete this assignment. All of the submissions will be compared with MOSS to determine if students have excessively collaborated, or have used the work of past students.
Any failure to respect the class rules, as explained above or in the syllabus, or the UC Davis Code of Conduct will automatically result in the matter being transferred to Student Judicial Affairs.