CMP2003 Data Structures and Algorithms (C++)
Term Project

-- Log Analyzer --

1. Main Requirements

You are expected to write a C++ console application which reads a text file that consists of logs of a web server. An example part of the log file is given below:

Host	TimeStamp	Filename	HTTP Reply Code	Bytes in Reply
local	- - [24/Oct/1994:13:41:41 -0600]	"GET index.html HTTP/1.0"	200	150
local	- - [24/Oct/1994:13:41:41 -0600]	"GET 1.gif HTTP/1.0"	200	1210
local	- - [24/Oct/1994:13:43:13 -0600]	"GET index.html HTTP/1.0"	200	3185
local	- - [24/Oct/1994:13:43:14 -0600]	"GET 2.gif HTTP/1.0"	200	2555
local	- - [24/Oct/1994:13:43:15 -0600]	"GET 3.gif HTTP/1.0"	200	36403
local	- - [24/Oct/1994:13:43:17 -0600]	"GET 4.gif HTTP/1.0"	200	441
local	- - [24/Oct/1994:13:46:45 -0600]	"GET index.html HTTP/1.0"	200	3185
…				
…				

After reading and processing is over, your program must list the “top 10” most visited web pages.

Sample output :

Filename1 # of total visits
Filename2 # of total visits
Filename3 # of total visits
.
.
.
Filename10 # of total visits

Total Elapsed Time : X seconds

You should extract the filenames from each line. For example, the first line in the above log file contains the filename “index.html” and the second line contains the filename “1.gif”.

The application can be implemented with console facilities (you do not need advanced GUI elements).

You are required to do the following tasks:

Task1: You must implement a hash table as your main data structure. You should implement your own hash table, you cannot use a hash table implementation from a library. This implementation should be a proper C++ class. You should use this hash table for storing the filenames and the corresponding number of visits of that page (filename). There is no restriction on the choice of the hash function and collision resolution method.

Task2: Use std::unordered_map data structure for storing the filenames and the corresponding number of visits.

Task 3: In order to compare the efficiency of your own hash table implementation and std::unordered_map, write a program which measures the total time it takes starting from reading access_log file until the end of printing the top 10 most visited pages. There is no need to ask user input.

Note1: For finding the top 10 most visited pages try to find an efficient method. Hint: you can use the heap data structures.

Note2: Other than the hash table implementation, you can use C++ standard library for your needs.

2. Submission

You are expected to submit:

1) The source code in a zip file.
2) A report (in PDF) containing the following details:
   a) The program output with total elapsed time as a screenshot. One screenshot for the run which uses our own hash table implementation and one screenshot for the run which uses std::unordered_map.
   b) Detailed explanation of your hash table implementation and the method of finding the top 10 most visited pages. For the hash table implementation: data structures, collusion resolution method, and hash function used should be explained. For the top 10 method: the data structures and the algorithms used should be explained.

The project is at most 5 PERSON size.
1 person project will decrease your grade by 20%.

The deadline is set 25 December 2023 11:59 pm. Submit your files from itslearning system.
Late submissions will get lower grade by 10% for each day.

3. Cheating Policy.

You are not supposed to use each other's source code. Also please do not use source code from the Internet, another person or your book's examples.

All the source codes will be filtered through a similarity analysis tool, which is known to be effective against many types of code copying and changing tricks. These projects will be graded as 0.

4. Evaluation

The evaluation criterion will be the amount of your work you did and your understanding of the concepts involved.

Any lack of 2 items mentioned above (Submission) may cause you to get low grades.

5. Bonuses

You can get bonuses for extra efforts :

* Good coding styles and OO programming skills

* Different implementation of collision resolution methods and their comparison.

* Trying to develop and use data structures and algorithms which will lead to the fastest running times.

* Or any other nice feature you can think of.

Clearly explain such extra efforts.