COMP 322
Winter Semester 2025

INSTRUCTOR: DR. CHAD ZAMMAR
chad.zammar@mcgill.ca

Assignment 1: Exploring Functions, Files, and Arrays.

Due date: 13 February 2025, 11:59 PM.

Before you start:

- Researching similar problems on the Internet is recommended. However, your submission should reflect your work and effort.
- Due to our limited time, some topics may not be covered in class. You are encouraged to find answers online, and you can also reach out to the TAs for guidance.
- Please submit your assignment before the deadline to avoid penalties or risk rejection.
- Submit one file called **assignment1.cpp** containing all the functions together with their implementations. It will also include the main() function that runs everything.

Make sure your code is clear and readable. **The readability of your code and the quality of your comments will be graded**.

- No submission by email. Submit your work to mycourses.
- If your code does not compile it will not be graded.
- For any assignment-related question, feel free to ask in the appropriate Ed discussion thread that is created for this purpose. Please don't reveal any solution or long code snippet in the discussion thread.

Objective:

The objective of this assignment is to design and implement a basic DNS routing system using the C++ programming language. DNS is a crucial component of the internet infrastructure, responsible for translating human-readable domain names (e.g., www.example.com) into the corresponding IP addresses that computers use to communicate with each other.

What is DNS?

DNS is a hierarchical and distributed naming system that maps domain names to IP addresses. It is essential for the functioning of the internet, as it allows users to access websites, email servers, and other internet-based services using easy-to-remember domain names instead of complex IP addresses.

Let's break down the DNS service into multiple components and we'll implement each component as a function.

Q1: Get DNS information for a specific domain [10 pts]

Implement a function called **get_ip_address_from_file** that takes a domain name as input and returns the corresponding IP address.

The signature of the function should be:

string get_ip_address_from_file(const string& domain_name)

The function should follow these steps:

1. Open the DNS file in read mode (the file is called dns.txt). We assume that dns.txt is present in the same folder where your CPP executable is so we don't need to deal with the full path.
2. Check if the file was opened successfully. If not, return an empty string.

3. Read the file line by line. For each line:
    a. Split the line into key-value pairs using a delimiter (e.g., '=').
    b. Return the corresponding value (the IP address) if the key is found.
4. If the key is not found, return an empty string to indicate that the IP address could not be found.
5. Close the file.

Here is a sample DNS text file containing a few domain name mappings to IP addresses:

www.example.com=192.168.1.100

www.google.com=8.8.8.8

www.github.com=192.30.255.113

www.stackoverflow.com=151.101.1.69

Q2: Improving performance with caching [20 pts]

Implement a caching mechanism to improve the performance of the get_ip_address_from_file function.

Background:

In the previous question, you implemented the get_ip_address_from_file function, which retrieves the IP address corresponding to a given domain name by loading the DNS information from a file. While this implementation works, it may not be efficient for applications that need to frequently look up IP addresses, as the file needs to be read and parsed every time the function is called.

To improve the performance of the get_ip_address_from_file function, you will implement a caching mechanism that stores the previously looked-up IP addresses in memory.

Requirements:

1. Create a new function called **get_ip_address**. It first checks if the requested domain name is present in the cache. If the domain name is found in the cache, return the corresponding IP address. The signature of the function should be:

   string get_ip_address(const string& domain_name)

2. If the domain name is not found in the cache, call the original get_ip_address_from_file function to load the DNS information from the file, and then store the domain name-IP address pair in the cache for quick future retrieval.
3. You may use the unordered_map data structure as a cache and declare it as global: **unordered_map<string, string> dnsMap;**

Q3: Utility functions [10 pts]

Implement a function that displays the content of the Cache on the screen. The signature of the function is:

void print_cache(void);

Implement another function that displays the content of the DNS file on the screen. The signature of the function is:

void print_dns_file(string filename);

These functions are very useful when debugging.

Q4: Cache Eviction Mechanism [20 pts]

In the previous question, you implemented a caching mechanism to improve the performance. However, the cache size needs to be limited to prevent it from consuming too much memory. You will now implement a cache eviction mechanism to remove the least recently used (LRU) entries when the cache reaches its maximum capacity. Let's consider that the cache can only hold a maximum of 5 entries.

Requirements:

1. Modify the cache data structure to store the domain name-IP address pairs, along with a way to determine either the time or the order in which the pair was added.
2. Implement an LRU eviction policy to remove the least recently used entries when the cache reaches its maximum capacity.
3. When an element is being accessed, its eviction status should change.
4. Feel free to use helper functions to make your code maintainable and readable.

Q5: Cache Clean-up Mechanism [20 pts]

the cache may become stale if the underlying DNS information changes, such as when domain names are removed or their IP addresses are updated in the DNS file.

You will now implement a mechanism to force the clean-up of the cache by removing entries that were deleted from the DNS file and updating entries that were modified.

Requirements:

Implement a function called **`clean_up_cache`** that performs the following tasks:

- Read the DNS information from the file using the **get_ip_address** function.
- Compare the domain name-IP address pairs in the cache with the ones in the DNS file.
- Remove any entries from the cache that are not present in the DNS file (i.e., the domain name has been deleted).
- Update any entries in the cache where the IP address in the DNS file is different from the one in the cache (i.e., the domain name's IP address has been updated).
- The signature of the function is **void clean_up_cache(string filename);**

Q6: Add or Update DNS records [20 pts]

Implement a function to let the user add a new record or update an existing record in the DNS file. Remember that when updating, you need to **update the Cache** as well in case that updated entry was already in the Cache.

The signature of the function is **void add_update(string filename);**

Requirements:

1. The function should ask the user to input a domain name and an IP address.
2. If the provided domain name already exists in the DNS file, then update its IP in the DNS file according to the user's input.
3. If the provided domain does not exist in the DNS file, add it as a new entry to the DNS file.

Use this main function for testing:

```
int main() {
    // Fill the cache to its maximum capacity
    std::cout << "IP address for www.example.com: " <<
get_ip_address("www.example.com") << std::endl;
    std::cout << "IP address for www.google.com: " <<
get_ip_address("www.google.com") << std::endl;
    std::cout << "IP address for www.github.com: " <<
get_ip_address("www.github.com") << std::endl;
    std::cout << "IP address for www.stackoverflow.com: " <<
get_ip_address("www.stackoverflow.com") << std::endl;
    std::cout << "IP address for www.reddit.com: " <<
get_ip_address("www.reddit.com") << std::endl;

    // Print the cache contents
    print_cache();
```

```
    // Access an existing entry to update its position in the LRU list
    std::cout << "IP address for www.google.com: " <<
get_ip_address("www.google.com") << std::endl;

    // Add a new entry, which should evict the least recently used entry
    std::cout << "IP address for www.twitter.com: " <<
get_ip_address("www.twitter.com") << std::endl;

    // Print the cache contents again to verify the eviction
    print_cache();

    // Modify the DNS file to simulate changes
    std::ofstream dnsFile("dns.txt", std::ios::trunc);
    dnsFile << "www.example.com=192.168.1.101" << std::endl;
    dnsFile << "www.google.com=8.8.8.8" << std::endl;
    dnsFile << "www.github.com=192.30.255.113" << std::endl;
    dnsFile << "www.stackoverflow.com=151.101.1.69" << std::endl;
    dnsFile << "www.reddit.com=151.101.1.70" << std::endl;
    dnsFile.close();

    // Clean up the cache
    clean_up_cache("dns.txt");

    // Print the cache contents again to verify the cleanup
    print_cache();

    // Print the contents of the DNS file
    print_dns_file("dns.txt");

    // Test the add_update function
    add_update("dns.txt");

    // Print the cache and DNS file contents again to verify the changes
    print_cache();
    print_dns_file("dns.txt");

    return 0;
}
```