Systems Software
Lab 6: Using the Standard Template Library (STL)

Lab Objectives

In this activity, students should demonstrate the following abilities:

1. Use the template classes **`list`**, **`map`**, and **`unordered_map`** from STL
2. Use the template function **`find`** from STL
3. Compare the performance of the search operations of the three data structures (linked list, binary search tree, and hash table)

Lab Assignment

In this lab, you will use the three data structures, **`list`**, **`map`**, and **`unordered_map`** from the STL, and compare the performance of their search operation.

1. Create the class **`Student`** as seen in class with the data members **`id`**, **`name`**, and **`gpa`**. The class has two constructors, getters and setters for every data member, and overloading operator functions for the following operators: **==**, **>**, **<**, **>>**, and **<<**.

2. Write a C++ program that creates three instances of the classes **`list`** , **`map`**, and **`unordered_map`** for the type **`Student`** . For classes **`map`** and **`unordered_map`**, use type **`int`** for the key (student id). The program reads in the text file **`students.txt`** and loads the students' information in the three data structures using the method **`push_back`** for **`list`**, and the **`[] operator`** for **`map`** and **`unordered_map`**.

3. Create an array of 10 student IDs and initialize it to the following IDs: `{91242, 87351, 13385, 55555, 37867, 98296, 22222, 62985, 33333, 48851}`

4. Search for each student id, from the list above, in the three data structures. Display the execution time of the search operation on each data structure for each student id.

5. Submit your program files (`student.h`, `student.cpp`, and `main.cpp`) in a zipped folder named **lab6**.

Note 1: Both **map** and **unordered_map** data structures have a member function **find()** to perform the search operation. However, **list** does not have such a member function. Therefore, for **list**, you should use the generic function **find()** from the library **algorithm**.

Note 2: To calculate the execution time of the search method, call method **clock()** before and after the statement that calls the search method and store the return values of **clock** in two variables **start** and **end** of type **clock_t**. The execution time is then calculated as **end-start**. **clock** returns the number of clock cycles of the program at the time **clock** is called.

Here is a code snippet that shows how to measure the execution time of the search operation in the linked list data structure.

```
clock_t start, end;
start = clock();
find(studentLL.begin(), studentLL.end(), s);//s is a Student object
end = clock();
double time = double (end - start);
cout << time << endl;
```