# LAB 6 – STL

ELEC 3150 – Object Oriented Programming (Fall 2023)

Nick Cebula

HW question 1:

Assumptions:

## Question 1)

- Write a program to simulate queue system in a supermarket
  - Customer will be waiting in a queue until they get served.
  - There are 2 registers. Each register will serve one customer in each loop
  - A new customer will be added to queue at every beginning of a loop
  - ***Note:*** Customer is a class (Attribute: name, Method: show_name())
  - Assuming, Currently there are 7 customers waiting in line as follow and each person will spend all of their money
    - David - $50, Tim - $30, John - $70, Pete - $55, Ann - $82, Beth - $90, Rob - $20
    - New customer will spend $30
  - Print the money that each register collect at the end of the day

Main.cpp:

```cpp
#include <iostream>
#include <vector>
#include "Customer.h"

using std::vector;
using std::cout;
using std::endl;

int main() {
    vector<Customer> line;
    vector<int> spending = { 50, 30, 70, 55, 82, 90, 20};
    line.push_back(Customer("David")); //50
    line.push_back(Customer("Tim"));   //30
    line.push_back(Customer("John"));  //70
    line.push_back(Customer("Pete"));  //55
    line.push_back(Customer("Ann"));   //82
    line.push_back(Customer("Beth"));  //90
    line.push_back(Customer("Rob"));   //20
    line.push_back(Customer());        //30

    int register1 = 0;
    int register2 = 0;
    int count = 0;
```

```cpp
while (count < spending.size()) {
    for (int i = 0; i < line.size(); i++) {
        cout << line[i].show_name() << " ";
    }
    cout << endl;
    // Register 1
    register1 += spending[count];
    cout << "Register 1: " << line[0].show_name() << " spends $" << spending[count] << endl;
    cout << "Register 1 cash is $ " << register1 << endl;
    // Delete first in line
    line.erase(line.begin());
    if (line.empty()) {
        break;
    }
    cout << "-----" << endl;
    count++;
    line.push_back(Customer());        //30
    spending.push_back(30);

    // Register 2
    if (count < spending.size()) {
        register2 += spending[count];
        cout << "Register 2: " << line[0].show_name() << " spends $" << spending[count] << endl;
        cout << "Register 2 cash is $ " << register2 << endl;
        // Delete first in line
        line.erase(line.begin());
        if (line.empty()) {
            break;
        }
    }
    count++;
}

cout << "Total money collected at Register 1: $" << register1 << endl;
cout << "Total money collected at Register 2: $" << register2 << endl;

return 0;
}
```

Customer.h:

```h
#pragma once
#include <string>
using std::string;
class Customer
{
    //attribute
    string name;
public:
    //constructor
    Customer();
    Customer(string in_name);
    //method
    string show_name();

    //destructor
    ~Customer();
    };
```

customer.cpp:

```cpp
#include "Customer.h"
//constructor
Customer::Customer() {
    name = "Default";
}
Customer::Customer(string in_name) {
    name = in_name;
}
//method
string Customer::show_name() {
    return name;
}

//destructor
Customer::~Customer() {

}
```

Description: I started off by making the customer class which was simple, its as a basic class that shows name. In the main.cpp started off by making a vector for the people in the line. Then another vector for their spendings. This is done to the lab assumptions because you did not ask for a attribute of cash for the customer. I then used line.pushback to set the people in line and their names. I made variables for both registers. I made a while loop to keep running till all the customers are served. Register 1 serves first adds the customer cash to its total and deletes the customer using erase function. It adds to count so it moves on to the next customer for each register and adds a new default customer once a day. At the end of the program it prints the total money collected for each register.

Lab Question 1:

Assumptions:

In this lab, you have to create a person class. This person class has the following attributes and methods.

Attributes:

- Name, job title, division/department

Methods:

- Constructors/destructor
- print_out
  - Print all attributes
- Get_name
  - Return name of this person
- Get_jobtitle
  - Return job title of this person
- Get_department
  - Return department of this person

We will be creating a **list** that can:

- Add new objects
- Search for a matching input (any of the attributes)
- Print all object in a list
- Remove a particular person

You will populate an STL structure with a series of person objects (at least 5).

Use the STL (standard template library) for a ***list***. Use the STL functions. You will need to read and understand the documentation for the library and report any site you use **(cite in the documentation or code comments)**. Code without citations will be penalized.

Main.cpp:

```cpp
#include <iostream>
#include <string>
#include "Person.h"
#include <list>
using std::cin;
using std::cout;
using std::string;
using std::endl;
using std::list;
using std::find;
#include <algorithm>

int main() {
    int choice = 0;
    // create list
    list<Person> people;
    people.push_back(Person("Nick", "Student", "Computer Eng."));
    people.push_back(Person("Elon Musk", "CEO", "Tesla"));
    people.push_back(Person("Messi", "Athlete", "Soccer"));
    people.push_back(Person("The Rock", "Actor", "Movie"));
    people.push_back(Person("Batman", "Hero", "DC"));
    cout << "Please select one of the following options: " << endl;
    cout << "1 - Print current list" << endl;
    cout << "2 - Add a new user to this list" << endl;
    cout << "3 - Search for a name" << endl;
    cout << "4 - Remove a person from this list" << endl;
    cout << "5 - Exit" << endl;
    cin >> choice;
```

```cpp
    while (choice >= 1 && choice <= 5) {        // PJ

        if(choice==1){
            for (auto person : people) {
                person.print_out();
            }
        }
        if(choice==2){
            string new_name;
            string new_job;
            string new_department;
            cout << "Enter name: ";
            cin >> new_name;
            cout << "Enter occupation: ";
            cin >> new_job;
            cout << "Enter department: ";
            cin >> new_department;
            people.push_back(Person(new_name, new_job, new_department));
            cout << "User added." << endl;
        }
        if(choice==3){
            string name;
            cout << "Enter name you are looking for: ";
            cin >> name;
            auto searching = find_if(people.begin(), people.end(), [name](Person& person) {
                return person.get_name() == name;
                });

            if (searching != people.end()) {
                cout << " " << name << " is in our system" << endl;
            }
            else {
                cout << "Person not found." << endl;
            }
        }
```

```
    if(choice==4){
        string remove;
        cout << "Enter a name who you want to remove: ";
        cin >> remove;
        auto remove_name = find_if(people.begin(), people.end(), [remove](Person& person) {
            return person.get_name() == remove;
            });
        if (remove_name != people.end()) {
            people.erase(remove_name);
            cout << " " << remove << " is removed from our system." << endl;
        }
        else {
            cout << "Person not found." << endl;
        }
    }
    if(choice==5){
        cout << "Exiting Program";
        break;
    }


    // PJ
    cout << "Please select one of the following options: " << endl;
    cout << "1 - Print current list" << endl;
    cout << "2 - Add a new user to this list" << endl;
    cout << "3 - Search for a name" << endl;
    cout << "4 - Remove a person from this list" << endl;
    cout << "5 - Exit" << endl;
    cin >> choice;
}

return 0;
}
```

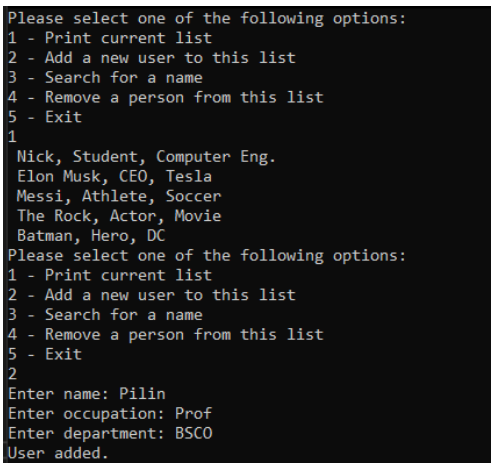
person.h:

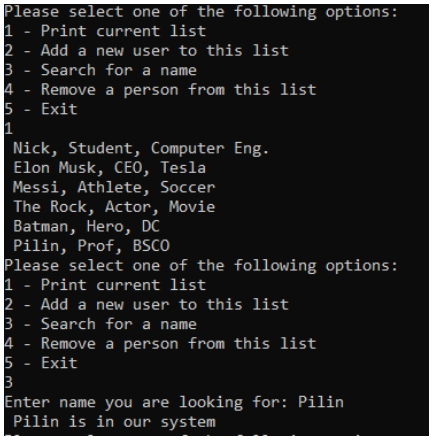
```
#pragma once
#include <string>
using std::string;
class Person
{
    //attributes
    string name;
    string job_title;
    string department;
    //constructor
public:
    Person();
    Person(string in_name, string in_job_title, string in_department);
    //methods
    void print_out();
    string get_name();
    string get_job_title();
    string get_department();
    //destructor
    ~Person();
};
```

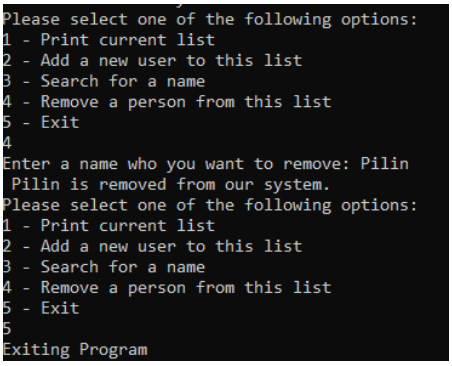
Person.cpp:

```
#include "Person.h"
#include <iostream>
#include <string>
using std::string;
using std::cout;
using std::endl;


Person::Person(string in_name, string in_job_title, string in_department) {
    name = in_name;
    job_title = in_job_title;
    department = in_department;
}


void Person::print_out()
{
    cout << " " << name << ", " << job_title << ", " << department << endl;
}

string Person::get_name()
{
    return name;
}

string Person::get_job_title()
{
    return job_title;
}

string Person::get_department()
{
    return department;
}

Person::~Person()
{
}
```

Results:




Description: I started off by making the person class, which was easy it helps with the printing values of the charactersitcs of the persons. Next, I used google to research the list library, which is found In my citations. I made a list which refers to the person class and named it ‘people’. Then I used people.pushback with in parenthesis the persons occupation and department. Pushback adds it to the list because its not like an array where you have a set amount of array enterences. Then I printed the list and let the use select what they want to do. For choice one it will print the people in the list. This uses auto which is used to call a varaible of a complicated type, it automatically determines this. For choice 2 is adds a new user using pushback and adds them to the end of the list. For option 3 it uses find_if function which is used to find a person in a list. It also usees the list.begin/end to search from start to end and sets it as a variable and checks verson the word that we are looking for using if statement. For choice 4 the remove, it takes user input and works the same exact way the search function works, which is very useful and fast but the only difference is it uses the erase function to remove. Choice 5 simply exits.

Citations: https://www.geeksforgeeks.org/list-cpp-stl/

https://cplusplus.com/reference/list/list/erase/

https://www.geeksforgeeks.org/std-find-in-cpp/

https://www.programiz.com/cpp-programming/list

https://stackoverflow.com/questions/3434256/use-the-auto-keyword-in-c-stl

Self-assessment: This lab was very intresting, not knwoing how to use the list library, I was able to google and learn how to use the included functions. It gave me confidence that I can do a lot of projects. It was challenging don't get me wrong but after figuring how to use the items, it was staright forward and the functionality was very useful for future projects. As you know I had problems with the update conditions which  was a simple mistake that I learned in microcontrollers, and this problem I was overthinking, it was simply a mistake with the WHILE loop and the conditions for user choice it would exit right away because I did not have the right logicals which I didn't realize before trying to add the update conditions in different places.