FINAL PROJECT

LAB MATERIALS BORROWING MANAGER (QUEUES)	
Course Code: CPE 010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:** December 2, 2024
Section: CPE21S4	**Date Submitted:** December 2, 2024
Name(s): - Adia, James Russel E. - Carag, Carl Jervie B. - Gob, Mark Jeonel Kenn E. - Guariño, Danica T. - Rio, Aries C.	**Instructor:** Engr. Maria Rizette H. Sayo

1. Objectives(s)

General Objectives:

- Follow a specific data structure to be implemented:
  - Queues
- The created output can be related to Computer Engineering

Specific Objectives:

- The program will be done using C++ programming language
- The program will utilize queues as the data structure for processing the borrower's request
- The interface of the program and its output will be in a Console Window

2. Intended Learning Outcomes (ILOs)

- **Develop a Console-Based Application:** Students will learn how to design and implement a console-based application with an interface.
- **Gain experience with file handling for data storage**: Students will develop skills in saving and loading data from text files to maintain records of users, materials, and borrowing requests.
- **Learn to manage and process user requests using queues**: Students will understand how to manage and process borrowing requests in a sequential manner, including the approval and denial of requests.

3. Discussion

I. Background and Concepts

Nowadays, T.I.P. students use the facilities that enable them to learn and simulate experiments that require the usage of equipment. These pieces of equipment are crucial in order for the students to perform their activities and tasks accordingly. The students must first go to the tool room, which has special designations for each department they are assigned to provide the necessary materials. The time it takes for the students to borrow each material is critical for the remaining time for them to accomplish their tasks. The Lab Materials Borrowing Manager enables them to reserve their needed materials on a personal computer. This application makes every student have an equal opportunity with the reservation system that allows first come, first served to the students to each material in the tool room.

This program uses queues data structure type that store multiple elements in a specific order, which is called FIFO or First In First Out, which has a principle that the first element stored comes out first. The Methods associated with this data structure are push() that inserts an element at the back of the queue, pop() removes an element from the front of the queue, front() returns the first element of the queue, back() returns the last element of the queue, size() returns the number of elements in the queue, empty() returns if the queue is empty. The system of the queue works by pushing the first element and then using the pop() method to remove that element from the list. We may access the elements inside the queue such as using front() that brings back the element at the front of the queue, and back() that places the element at the back of the queue. Checking if the queue has elements on it uses the empty() method. If the statement is true then the queue is considered empty, if the statement was false then the queue is not empty.

The <iostream> library is an object-oriented library that allows the input and output functionality of the code to operate. The <fstream> is a preprocessor that includes the necessary classes and functions for files in C++, allowing the program to read and write files. The <sstream> header file converts string to create a StringStream object; it associates a string object with a stream that allows you to read the string as if it were a stream. The <map> in C++ is part of the STL, or Standard Template Library; it stores elements in a mapped fashion; these elements have a key value and a mapped value. The <vector> header file is part of the STL, or Standard Template Library; it allows implementation of a vector function, which is a dynamic array that has the ability to change the size of itself exactly when an element is inserted or deleted. The <string> header file contains the std::string class, which is a standard representation for a text string, which includes common string operations.

The <queue> is also part of the STL, or Standard Template Library, which allows the insertion, removing, and accessing of an element using push(), pop(), front(), back(), and empty().

II. Scope and Limitations

The Lab Materials Borrowing Manager program allows students to register, log in, and request lab materials, while admins can manage the available materials and process borrowing requests. The program handles the storage of user accounts, material data, and pending borrowing requests through text files, ensuring persistent data storage. Students can specify the materials and quantities they wish to borrow, and the admin has the authority to approve or deny these requests. The program is designed to provide a simple and efficient system for managing lab material borrowing in a university or lab setting.

The program has some limitations, such as using text files to store data, which can cause problems if there are many users or if multiple people try to use the system at the same time. It only supports one admin with basic login credentials, and it doesn't offer strong security, like passwords for users. The program doesn't reserve materials in real-time. Additionally, it has no graphical user interface and could improve in handling errors and providing status updates to students about their borrowing requests.

4. Materials and Equipment

- A computer or laptop with an operating system that supports C++ development (Windows, macOS, or Linux)

- A C++ compiler with version (C++11 or later) such as:
  - GCC (GNU Compiler Collection)
  - Clang
  - Microsoft Visual C++ (MSVC)

- An IDE such as:
  - Visual Studio Code (VSCode - Follow VSCode documentation on how to setup C++)
  - CLion
  - Code Blocks
  - Visual Studio
  - Embarcadero DevC++

5. Procedure

ILO A: Develop a Console-Based Application

ILO B: Gain experience with file handling for data storage

ILO C: Learn to manage and process user request using queues

We would first need to include the necessary header files for the program and use the standard namespace

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <map>
#include <vector>
#include <string>
#include <queue>

using namespace std;
```

Note: The following are only pseudocodes to guide the coding process

Create an Material class that will handle the names and quantity of the materials that will be provided

```
C/C++
Class Material
    Properties
        String name
        Integer quantity

    Constructor Material(name, quantity)
        this.name = name
        this.quantity = quantity
End Class
```

Create an Account Manager class that will handle the different account information and the process of storing it into a accounts.txt file

```
C/C++
Class AccountManager
    Properties
        String filename
```

```
    Constructor AccountManager(filename = "accounts.txt")
        this.filename = filename

    Method registerAccount(name, studentNumber)
        Open file for reading filename
        While reading each line from file
            If existingName == name OR existingNumber == studentNumber
                Return false  // User already exists
        End While

        Open file for appending filename
        Write name, studentNumber to file
        Return true  // Registration successful

    Method login(name, studentNumber)
        Open file for reading filename
        While reading each line from file
            If existingName == name AND existingNumber == studentNumber
                Return true  // Login successful
        End While
        Return false  // Credentials incorrect
End Class
```

Create a Database Manager class that will handle the different information of the materials and the process of storing the materials into a database.txt file

```
C/C++
Class DatabaseManager
    Properties
        String filename
        Map<String, Integer> materials

    Constructor DatabaseManager(filename = "database.txt")
        this.filename = filename
        Call loadMaterials()

    Method loadMaterials()
        Open file for reading filename
        While reading each line from file
            Parse name and quantity
            Store name with quantity in materials map

    Method saveMaterials()
        Open file for writing filename
        For each material in materials
```

```
            Write material.name, material.quantity to file

    Method addMaterial(name, quantity)
        Increment materials[name] by quantity
        Call saveMaterials()

    Method removeMaterial(name)
        Remove name from materials
        Call saveMaterials()

    Method getMaterials()
        Return materials
End Class
```

Create a Borrowing Request class that will handle the requests of the students, this will also be the information wherein the concept of queues will be applied

```
C/C++
Class BorrowingRequest
    Properties
        String studentName
        String studentNumber
        List<Material> requestedMaterials

    Constructor BorrowingRequest(name, number)
        this.studentName = name
        this.studentNumber = number
End Class
```

Create the Borrowing App class that will include all the functionalities needed in the program. This will also be the class wherein the formatting of the program that will be seen in the console window can be modified. This class would also include the method that will handle the queuing process of the requests. Overall this would be the main driver code for the program.

```
C/C++
Class BorrowingApp
    Properties
        DatabaseManager dbManager
        Queue<BorrowingRequest> borrowingQueue
        LogFile logFile
        String pendingRequestsFile = "pendingRequests.txt"
```

```
    Constructor BorrowingApp()
        Call loadPendingRequests()

    Destructor ~BorrowingApp()
        Call savePendingRequests()

    Method loadPendingRequests()
        Open pendingRequestsFile for reading
        While reading each line from file
            Create BorrowingRequest from line data
            Push request to borrowingQueue

    Method savePendingRequests()
        Open pendingRequestsFile for writing
        While borrowingQueue is not empty
            Write request data to file

    Method start()
        While true
            Display main menu
            Read choice
            Execute corresponding method (studentLogin, studentRegister, adminAccess,
exit)

    Method studentLogin()
        Read student name and number
        If valid credentials
            Create BorrowingRequest
            Call handleBorrowingRequest(request)

    Method studentRegister()
        Read student name and number
        Validate student number
        Call AccountManager.registerAccount()

    Method adminAccess()
        Authenticate admin
        Display admin menu
        Execute corresponding method based on choice

    Method handleBorrowingRequest(request)
        While true
            Display available materials
            Read material name and quantity
            If "done"
                Call finishBorrowing(request)
                Break
            Call addMaterial(request, materialName, quantity)

    Method addMaterial(request, materialName, quantity)
```

```
        Check availability in dbManager
        Add to request and update dbManager materials

    Method processAdminRequests()
        While borrowingQueue is not empty
            Process top request
            Prompt admin for decision
            Log decision and save changes

    Method finishBorrowing(request)
        If no materials requested
            Inform user
        Add request to borrowingQueue
        Log borrowing submission
End Class
```

Create the main function that will let the program to be executed

```
C/C++
Function main()
    Create instance of BorrowingApp
    Call start() on app instance
End Function
```

6. Output

I. Source Code

```
C/C++
#include <iostream>
#include <fstream>
#include <sstream>
#include <map>
#include <vector>
#include <string>
#include <queue>

using namespace std;
```

```cpp
/* Class for the Materials in the Laboratory */
class Material {
public:
    string name;
    int quantity;

    Material(const string& name, int quantity) : name(name), quantity(quantity) {}
};

/* Class for the Account Manager */
class AccountManager {
private:
    string filename;

public:
    AccountManager(const string& filename = "accounts.txt") : filename(filename) {}

    bool registerAccount(const string& name, const string& studentNumber) {
        ifstream file(filename);
        string line;

        while (getline(file, line)) {
            istringstream ss(line);
            string existingName, existingNumber;
            getline(ss, existingName, ',');
            getline(ss, existingNumber);
            if (existingName == name || existingNumber == studentNumber) {
                return false;  // User already exists
            }
        }

        ofstream outFile(filename, ios::app);
        outFile << name << "," << studentNumber << "\n";  // Save new user
        return true;  // Registration successful
    }

    bool login(const string& name, string& studentNumber) {
        ifstream file(filename);
        string line;

        while (getline(file, line)) {
            istringstream ss(line);
            string existingName, existingNumber;
            getline(ss, existingName, ',');
            getline(ss, existingNumber);
            if (existingName == name && existingNumber == studentNumber) {
                return true;  // Login successful
            }
        }
        return false;  // Credentials incorrect
```

```
    }
};

/* Class for the Database Manager */
class DatabaseManager {
private:
    string filename;
    map<string, int> materials;

public:
    DatabaseManager(const string& filename = "database.txt")
                    : filename(filename) {
                    loadMaterials();
    }

    void loadMaterials() {
        ifstream file(filename);
        string line;

        while (getline(file, line)) {
            istringstream ss(line);
            string name;
            int quantity;
            getline(ss, name, ',');
            ss >> quantity;
            materials[name] = quantity;
        }
    }

    void saveMaterials() {
        ofstream file(filename);
        for (const auto& material : materials) {
            file << material.first << "," << material.second << "\n";
        }
    }

    void addMaterial(const string& name, int quantity) {
        materials[name] += quantity;  // Add to existing quantity
        saveMaterials();
    }

    void removeMaterial(const string& name) {
        materials.erase(name);
        saveMaterials();
    }

    const map<string, int>& getMaterials() const {
        return materials;
    }
};
```

```
/* Class for the Borrowing Request */
class BorrowingRequest {
public:
    string studentName;
    string studentNumber;
    vector<Material> requestedMaterials;

    BorrowingRequest(const string& name, const string& number)
        : studentName(name), studentNumber(number) {}
};

/* Class for the Borrowing Menu or App */
class BorrowingApp {
private:
    DatabaseManager dbManager;
    queue<BorrowingRequest> borrowingQueue;
    ofstream logFile;
    ofstream requestFile;
    string pendingRequestsFile = "pendingRequests.txt";

public:
    BorrowingApp() : dbManager(), logFile("log.txt", ios::app) {
        loadPendingRequests();
    }

    ~BorrowingApp() {
        savePendingRequests();
    }

    void loadPendingRequests() {
        ifstream file(pendingRequestsFile);
        string line;
        while (getline(file, line)) {
            istringstream ss(line);
            string studentName, studentNumber;
            getline(ss, studentName, ',');
            getline(ss, studentNumber, ',');
            BorrowingRequest request(studentName, studentNumber);
            while (getline(ss, line, ',')) {
                string materialName;
                int quantity;
                istringstream materialStream(line);
                getline(materialStream, materialName, ':');
                materialStream >> quantity;
                request.requestedMaterials.emplace_back(materialName, quantity);
            }
            borrowingQueue.push(request);
        }
    }
```

```
    void savePendingRequests() {
        ofstream file(pendingRequestsFile);
        queue<BorrowingRequest> tempQueue = borrowingQueue;
        while (!tempQueue.empty()) {
            BorrowingRequest request = tempQueue.front();
            tempQueue.pop();
            file << request.studentName << "," << request.studentNumber;
            for (const auto& material : request.requestedMaterials) {
                file << "," << material.name << ":" << material.quantity;
            }
            file << "\n";
        }
    }

    void start() {
        while (true) {
            cout << "\n-----------------------------------------------\n";
            cout << "Welcome to the Lab Materials Borrowing Interface!\n";
            cout << "-----------------------------------------------\n";
            cout << "Main Menu:\n";
            cout << "1. Borrower's Log In\n";
            cout << "2. Borrower's Registration\n";
            cout << "3. Admin Access\n";
            cout << "4. Exit\n\n";
            cout << "Enter choice (number only): ";

            int choice;
            if (!(cin >> choice)) {
                cin.clear();  // Clear the error flag
                cin.ignore(10000, '\n');  // Discard invalid input
                cout << "Invalid input. Please enter a number.\n";
                continue;
            }
            cout << "\n";
            cin.ignore(10000, '\n');  // Clear the newline character from the input
buffer

            if (choice == 1) {
                studentLogin();
            } else if (choice == 2) {
                studentRegister();
            } else if (choice == 3) {
                adminAccess();
            } else if (choice == 4) {
                break;
            } else {
                cout << "Invalid option. Please try again.\n";
            }
        }
```

```
}

void studentLogin() {
    cout << "----------------------\n";
    cout << "   Borrower's Login    \n";
    cout << "----------------------\n";
    string studentName, studentNumber;
    cout << "Enter your name (or 'return' to go back): ";
    getline(cin, studentName);

    if (studentName == "return") {
        return;  // Return to the main menu
    }

    cout << "Enter your student number: ";
    getline(cin, studentNumber);

    AccountManager accountManager;
    if (accountManager.login(studentName, studentNumber)) {
        BorrowingRequest request(studentName, studentNumber);
        handleBorrowingRequest(request);
    } else {
        cout << "Invalid credentials. Please try again.\n\n";
    }
}

void studentRegister() {
    cout << "----------------------------\n";
    cout << "   Borrower's Registration    \n";
    cout << "----------------------------\n";
    string studentName, studentNumber;
    cout << "Enter your name (or 'return' to go back): ";
    getline(cin, studentName);

    if (studentName == "return") {
        return;  // Return to the main menu
    }

    cout << "Enter your student number: ";
    getline(cin, studentNumber);

    // Check if student number is purely integers
    for (char c : studentNumber) {
        if (!isdigit(c)) {
            cout << "Invalid student number. Please enter numbers only.\n\n";
            return;
        }
    }

    AccountManager accountManager;
```

```
        if (accountManager.registerAccount(studentName, studentNumber)) {
            cout << "Registration successful!\n\n";
        } else {
            cout << "Registration failed. User already exists.\n\n";
        }
    }

    void adminAccess() {
        cout << "-----------------------------\n";
        cout << "         Admin Access    \n";
        cout << "-----------------------------\n";
        string adminName, adminPassword;
        cout << "Enter admin name (or 'return' to go back): ";
        getline(cin, adminName);

        if (adminName == "return") {
            return;  // Return to the main menu
        }

        cout << "Enter admin password: ";
        getline(cin, adminPassword);

        if (adminName == "Admin" && adminPassword == "admin") {
            while (true) {
                cout << "\nAdmin Menu:\n";
                cout << "1. View Materials\n";
                cout << "2. Add Material\n";
                cout << "3. Remove Material\n";
                cout << "4. Process Borrowing Requests\n";
                cout << "5. Exit Admin Acess\n\n";
                cout << "Enter choice (number only): ";

            int choice;
            if (!(cin >> choice)) {
                cin.clear();  // Clear the error flag
                cin.ignore(10000, '\n');  // Discard invalid input
                cout << "Invalid input. Please enter a number.\n\n";
                continue;
            }
            cout << "\n";
            cin.ignore(10000, '\n');  // Clear the newline character from the input
buffer

                if (choice == 1) {
                    viewMaterials();
                } else if (choice == 2) {
                    addMaterialAdmin();
                } else if (choice == 3) {
                    removeMaterialAdmin();
                } else if (choice == 4) {
```

```
                processAdminRequests();
            } else if (choice == 5) {
                break;
            } else {
                cout << "Invalid option. Please try again. \n\n";
            }
        }
    } else {
        cout << "Invalid admin credentials.\n\n";
    }
}

void viewMaterials(){
    cout << "\n--------------------\n";
    cout << "Available Materials\n";
    cout << "--------------------\n";
    for (const auto& material : dbManager.getMaterials()) {
        cout << material.first << ": " << material.second << "\n";
    }
}

void addMaterialAdmin() {
    viewMaterials();

    string materialName;
    int quantity;

    cout << "\nEnter material name to add (or 'return' to go back): ";
    getline(cin, materialName);

     if (materialName == "return") {
        return;  // Return to the admin menu
    }

    cout << "Enter quantity: ";

    while (!(cin >> quantity)) {
        cin.clear();  // Clear the error flag
        cin.ignore(10000, '\n');  // Discard invalid input
        cout << "Invalid input. Please enter a number: ";
    }
    cin.ignore(10000, '\n');  // Clear the newline character from the input buffer

    dbManager.addMaterial(materialName, quantity);
    cout << "Material added successfully.\n";
}

void removeMaterialAdmin() {
    viewMaterials();
```

```
        string materialName;
        cout << "\nEnter material name to remove (or 'return' to go back): ";
        getline(cin, materialName);

        if (materialName == "return") {
            return;  // Return to the admin menu
        }

        if (dbManager.getMaterials().count(materialName) == 0) {
            cout << "Material not found.\n";
            return;
        }

        dbManager.removeMaterial(materialName);
        cout << "Material removed successfully.\n";
    }

    void handleBorrowingRequest(BorrowingRequest& request) {
        while (true) {

            viewMaterials();

            string materialName;
            int quantity;

            cout << "\nEnter material name to borrow (or 'done' to finish): ";
            getline(cin, materialName);
            if (materialName == "done") {
                finishBorrowing(request);
                break;
            }

            cout << "Enter quantity: ";
            if (!(cin >> quantity)) {
                cin.clear();  // Clear the error flag
                cin.ignore(10000, '\n');  // Discard invalid input
                cout << "Invalid input. Please input requested item's information
properly.\n\n";
                continue;
            }
            cin.ignore(10000, '\n');  // Clear the newline character from the input
buffer

            addMaterial(request, materialName, quantity);
        }
    }

    void addMaterial(BorrowingRequest& request, const string& materialName, int
quantity) {
        const auto& materials = dbManager.getMaterials();
```

```
        if (materials.count(materialName) == 0) {
            cout << "Material not found.\n";
            return;
        }

        int availableQuantity = materials.at(materialName);
        if (quantity > availableQuantity) {
            cout << "Only " << availableQuantity << " available for " << materialName
<< ".\n";
            return;
        }

        request.requestedMaterials.emplace_back(materialName, quantity);
        dbManager.addMaterial(materialName, -quantity);  // Deduct the requested
quantity from the database
        cout << "Added " << quantity << " of " << materialName << " to your borrowing
request.\n";
    }


    void processAdminRequests() {
        cout << "You have " << borrowingQueue.size() << " pending requests.\n\n";
        if (!borrowingQueue.empty()) {
            BorrowingRequest request = borrowingQueue.front();
            borrowingQueue.pop();

            cout << "Processing request for " << request.studentName << "...\n";
            cout << "Materials Requested:\n";
            for (const auto& material : request.requestedMaterials) {
                cout << "- " << material.name << ": " << material.quantity << "\n";
            }

            string decision;
            while (true) {
                cout << "Approve this request? (y = yes/n = no): ";
                cin >> decision;
                cin.ignore(10000, '\n');  // Clear the newline character from the
input buffer
                if (decision == "y" || decision == "n") {
                    break;
                }
                cout << "Invalid input. Please enter 'y' or 'n'.\n";
            }

            logFile << "Processing request for: " << request.studentName << " (" <<
request.studentNumber << ")\n";
            logFile << "Materials Requested:\n";
            for (const auto& material : request.requestedMaterials) {
                logFile << "- " << material.name << ": " << material.quantity << "\n";
            }
```

```
            if (decision == "y") {
                cout << "Request approved.\n";
                logFile << "Decision: Approved\n";
            } else {
                for (const auto& material : request.requestedMaterials) {
                    dbManager.addMaterial(material.name, material.quantity);  //
Return the quantity back to the database
                }
                cout << "Request denied.\n";
                logFile << "Decision: Denied\n";
            }

            logFile << "-----------------------------------\n";
            savePendingRequests();  // Save the updated queue to the file
        } else {
            cout << "No pending requests.\n";
        }
    }

    void finishBorrowing(BorrowingRequest& request) {
        if (request.requestedMaterials.empty()) {
            cout << "No materials requested.\n\n";
            return;
        }

        borrowingQueue.push(request);  // Add the request to the queue

        logFile << "Borrowing Request Submitted:\n";
        logFile << "Borrower: " << request.studentName << " (" <<
request.studentNumber << ")\n";
        logFile << "Materials Requested:\n";
        for (const auto& material : request.requestedMaterials) {
            logFile << "- " << material.name << ": " << material.quantity << "\n";
        }
        logFile << "-----------------------------------\n";

        cout << "\nBorrowing Request Submitted:\n";
        cout << "Thank you for using the borrowing system! Please wait for your
request to be processed.\n\n";
    }
};

int main() {
    BorrowingApp app;
    app.start();
    return 0;
}
```

II. Outputs

Lab Materials Borrowing Interface:

```
------------------------------------------------
Welcome to the Lab Materials Borrowing Interface!
------------------------------------------------
Main Menu:
1. Borrower's Log In
2. Borrower's Registration
3. Admin Access
4. Exit

Enter choice (number only): 
```

Admin Access Console:

```
----------------------------
         Admin Access
----------------------------
Enter admin name (or 'return' to go back): Admin
Enter admin password: admin

Admin Menu:
1. View Materials
2. Add Material
3. Remove Material
4. Process Borrowing Requests
5. Exit Admin Acess

Enter choice (number only):
```

Add Materials (Admin):

```
Enter choice (number only): 2


--------------------
Available Materials
--------------------

Enter material name to add (or 'return' to go back): Resistor
Enter quantity: 47
Material added successfully.

Admin Menu:
1. View Materials
2. Add Material
3. Remove Material
4. Process Borrowing Requests
5. Exit Admin Acess

Enter choice (number only): 2


--------------------
Available Materials
--------------------
Resistor: 47

Enter material name to add (or 'return' to go back): Diode 25
Enter quantity: 25
Material added successfully.
```

View Materials (Admin):

```
Admin Menu:
1. View Materials
2. Add Material
3. Remove Material
4. Process Borrowing Requests
5. Exit Admin Acess

Enter choice (number only): 1


--------------------
Available Materials
--------------------
Diode 25: 25
Resistor: 47
```

Removing Materials (Admin):

```
Enter choice (number only): 3


--------------------
Available Materials
--------------------
Diode 25: 25
Resistor: 47

Enter material name to remove (or 'return' to go back): Diode 25
Material removed successfully.

Admin Menu:
1. View Materials
2. Add Material
3. Remove Material
4. Process Borrowing Requests
5. Exit Admin Acess

Enter choice (number only): 1


--------------------
Available Materials
--------------------
Resistor: 47
```

Borrower Registration:

```
Enter choice (number only): 2

----------------------------
   Borrower's Registration
----------------------------
Enter your name (or 'return' to go back): John Doe
Enter your student number: 12345
Registration successful!
```

Borrower Login:

```
----------------------
   Borrower's Login
----------------------
Enter your name (or 'return' to go back): John Doe
Enter your student number: 12345

-------------------
Available Materials
-------------------

Enter material name to borrow (or 'done' to finish):
```

Borrowing Materials:

```
Enter choice (number only): 1

----------------------
   Borrower's Login
----------------------
Enter your name (or 'return' to go back): John Doe
Enter your student number: 12345

-------------------
Available Materials
-------------------
Resistor: 47

Enter material name to borrow (or 'done' to finish): Resistor
Enter quantity: 12
Added 12 of Resistor to your borrowing request.

-------------------
Available Materials
-------------------
Resistor: 35

Enter material name to borrow (or 'done' to finish): done

Borrowing Request Submitted:
Thank you for using the borrowing system! Please wait for your re
quest to be processed.
```

Approving Borrow Request:

```
------------------------------------------------
Welcome to the Lab Materials Borrowing Interface!
------------------------------------------------
Main Menu:
1. Borrower's Log In
2. Borrower's Registration
3. Admin Access
4. Exit

Enter choice (number only): 3

----------------------------
         Admin Access
----------------------------
Enter admin name (or 'return' to go back): Admin
Enter admin password: admin

Admin Menu:
1. View Materials
2. Add Material
3. Remove Material
4. Process Borrowing Requests
5. Exit Admin Acess

Enter choice (number only): 4

You have 1 pending requests.

Processing request for John Doe...
Materials Requested:
- Resistor: 12
Approve this request? (y = yes/n = no): y
Request approved.
```

Exit Program:

```
------------------------------------------------
Welcome to the Lab Materials Borrowing Interface!
------------------------------------------------
Main Menu:
1. Borrower's Log In
2. Borrower's Registration
3. Admin Access
4. Exit

Enter choice (number only): 4



...Program finished with exit code 0
Press ENTER to exit console.
```

{{Error Message Cases}}

Invalid Option:

```
------------------------------------------------
Welcome to the Lab Materials Borrowing Interface!
------------------------------------------------
Main Menu:
1. Borrower's Log In
2. Borrower's Registration
3. Admin Access
4. Exit

Enter choice (number only): 5

Invalid option. Please try again.

------------------------------------------------
Welcome to the Lab Materials Borrowing Interface!
------------------------------------------------
Main Menu:
1. Borrower's Log In
2. Borrower's Registration
3. Admin Access
4. Exit

Enter choice (number only): 
```

Invalid Admin Access:

```
------------------------------------------------
Welcome to the Lab Materials Borrowing Interface!
------------------------------------------------
Main Menu:
1. Borrower's Log In
2. Borrower's Registration
3. Admin Access
4. Exit

Enter choice (number only): 3

-----------------------------
         Admin Access
-----------------------------
Enter admin name (or 'return' to go back): adminn
Enter admin password: Adminn
Invalid admin credentials.
```

Non-numerical Student Number Entry:

```
-----------------------------
   Borrower's Registration
-----------------------------
Enter your name (or 'return' to go back): Jane Doe
Enter your student number: asdfg
Invalid student number. Please enter numbers only.
```

Incorrect / Non-existent Login Credentials:

```
------------------------------------------------
Welcome to the Lab Materials Borrowing Interface!
------------------------------------------------
Main Menu:
1. Borrower's Log In
2. Borrower's Registration
3. Admin Access
4. Exit

Enter choice (number only): 1

----------------------
   Borrower's Login
----------------------
Enter your name (or 'return' to go back): Jane Doe
Enter your student number: 54321
Invalid credentials. Please try again.
```

```
Enter choice (number only): 1

----------------------
   Borrower's Login
----------------------
Enter your name (or 'return' to go back): John Doe
Enter your student number: 32415
Invalid credentials. Please try again.
```

Non-numerical Entry for Add Materials (Admin):

```
-------------------
Available Materials
-------------------

Enter material name to add (or 'return' to go back): Breadboard
Enter quantity: Asdfgh
Invalid input. Please enter a number: hello
Invalid input. Please enter a number:
```

Removing Non-existent Material:

```
-------------------
Available Materials
-------------------
Breadboard: 32

Enter material name to remove (or 'return' to go back): Bread
Material not found.
```

No Pending Requests:

```
Admin Menu:
1. View Materials
2. Add Material
3. Remove Material
4. Process Borrowing Requests
5. Exit Admin Acess

Enter choice (number only): 4

You have 0 pending requests.

No pending requests.
```

Material Not Found in Borrowing Request:

```
----------------------
   Borrower's Login
----------------------
Enter your name (or 'return' to go back): John Doe
Enter your student number: 12345

-------------------
Available Materials
-------------------
Breadboard: 32

Enter material name to borrow (or 'done' to finish): Beaker
Enter quantity: 6
Material not found.
```

No Materials Borrowed:

```
-------------------
Available Materials
-------------------
Breadboard: 32

Enter material name to borrow (or 'done' to finish): done
No materials requested.
```

7. Conclusion

I. Conclusion

The developed program successfully meets the outlined objectives by implementing a comprehensive borrowing system for laboratory materials, tailored for students that will need to borrow laboratory materials like Computer Engineering students. Utilizing the C++ programming language, the program effectively demonstrates the use of queues as the primary data structure for managing and processing borrower requests. This choice of data structure ensures that requests are handled in a first-in, first-out (FIFO) manner, which is ideal for maintaining an orderly and fair processing sequence. The program's interface and output are designed to operate within a console window, providing a straightforward and user-friendly experience for both students and administrators. Through this implementation, the program not only adheres to the specified data structure requirements but also showcases practical applications of C++ in solving real-world problems.

II. Recommendations

Some recommendations about our program that we can think of are the following:

- Security Enhancements: implement more security measures to protect sensitive data by requiring a password for each student
- User Experience Improvements: enhance user experience by providing clearer instructions and by displaying a confirmation message if the task done by the user is successful
- Implement the option for when students will return back the materials that they borrowed
- Implement a backup and recovery mechanism to protect against sudden data losses

8. References

Simplilearn. (2024, October 17). *What is StringStream in C++ with examples and Syntax*. Simplilearn.com. https://www.simplilearn.com/tutorials/cpp-tutorial/string-stream-in-cpp#:~:text=Using%20the%20header%20file,as%20an%20input%20to%20it.

C++ Queue (With examples). (n.d.). https://www.programiz.com/cpp-programming/queue

GeeksforGeeks. (2024, October 11). *Map in C++ Standard Template Library (STL)*. GeeksforGeeks. https://www.geeksforgeeks.org/map-associative-containers-the-c-standard-template-library-stl/

GeeksforGeeks. (2023b, October 31). *String functions in C++*. GeeksforGeeks. https://www.geeksforgeeks.org/cpp-string-functions/

GeeksforGeeks. (2024b, October 11). *Queue in C++ Standard Template Library (STL)*.

GeeksforGeeks. https://www.geeksforgeeks.org/queue-cpp-stl/