Due date: May 12, 11:59 PM

Project 3

Overview

This project creates a **Student Database Management System** that manages student and faculty records using a console interface (text-based menu). It supports adding, searching, updating, deleting, displaying, and sorting records for academic or administrative tasks. By using **polymorphism**, a single LinkedList<Person*> can store both Student and Faculty objects (since they inherit from Person), making the system more flexible and efficient.

Part 1: Abstract Base Class – Person

- **Purpose:** A base class for students and faculty, enabling polymorphism.
- **Member Variables:**
  - string name: The person's full name (e.g., "John Doe").
  - int id: A unique ID number for each person.
- **Member Functions:**
  - **Constructors:**
    - Default: Sets empty name and ID 0.
    - Parameterized: Takes name and id.
  - **virtual void display() = 0:** Pure virtual function to show details (overridden by child classes).
  - **virtual ~Person():** Virtual destructor to ensure proper cleanup of derived objects.
- **Files:** Person.h, Person.cpp

Part 2: Student Class (Inherits from Person)

- **Purpose:** Represents a student with specific details.
- **Private Members:**
  - string major: Field of study (e.g., "Computer Science").
  - double gpa: Grade point average (e.g., 3.5).
- **Public Member Functions:**
  - **Default Constructor:** Sets major to empty, gpa to 0.0, calls Person default constructor.
  - **Parameterized Constructor:** Takes name, id, major, gpa.
  - **Getters:** Get name, id, major, gpa.
  - **Setters:** Set major, gpa (name and ID via Person constructor).
  - **void display():** Overrides to print student info (e.g., "Student - ID: 101, Name: Jane, Major: Math, GPA: 3.8").
- **Files:** Student.h, Student.cpp

Part 3: Faculty Class (Inherits from Person)

- **Purpose:** Represents a faculty member with specific details.

- **Private Members:**
  - string department: Department (e.g., "Science").
  - string title: Job title (e.g., "Professor").
  - int salary: Yearly salary (e.g., 75000).
- **Public Member Functions:**
  - **Default Constructor:** Sets department and title to empty, salary to 0, calls Person default constructor.
  - **Parameterized Constructor:** Takes name, id, department, title, salary.
  - **Getters:** Get all member variables.
  - **Setters:** Set department, title, salary.
  - **void display():** Overrides to print faculty info (e.g., "Faculty - ID: 201, Name: Dr. Smith, Dept: Math, Title: Professor, Salary: $80000").
- **Files:** Faculty.h, Faculty.cpp

Part 4: Template Class – ListNode<T>

- **Purpose:** A node in the linked list to hold any type (now T will be Person*).
- **Members:**
  - T* data: Pointer to the stored object (e.g., Student* or Faculty* cast to Person*).
  - ListNode<T>* next: Pointer to the next node (or nullptr if last).
- **Files:** ListNode.h, ListNode.cpp

Part 5: Template Class – LinkedList<T> (Specialized for T = Person)*

- **Purpose:** A linked list to store Person* pointers, handling both Student and Faculty via polymorphism.
- **Private Member:**
  - ListNode<Person*>* head: Points to the first node (starts as nullptr).
- **Public Functions:**
  - **LinkedList():** Sets head to nullptr.
  - **~LinkedList():** Deletes all nodes and their Person* objects to free memory.
  - **void insert(Person* data):** Adds a new node with data (e.g., Student* or Faculty*) at the end.
  - **void display():** Calls display() on each Person* object (polymorphism decides the output).
  - **Person* search(int id):** Returns the Person* with the given id (or nullptr if not found).
  - **Person* search(string firstName, string lastName):** Finds by parsing name (or nullptr if not found).
  - **void update(int id):** Finds by id, prompts user to update fields (checks type to handle Student or Faculty specifics).

- ◦ **void remove(int id):** Deletes the node with the given id and its Person* object.
- **Files:** LinkedList.h, LinkedList.cpp

Part 6: Input Options – Add Students

- **Purpose:** Add students to the LinkedList<Person*>.
  1. **From Console:**
     - Prompt for id, name, major, gpa.
     - Create a Student object, insert as Person*.
  2. **From Text File (.txt):**
     - Format: 101 John CS 3.5.
     - Parse each line, create a Student, insert as Person*.
  3. **From CSV File (.csv):**
     - Format: 102,Jane,Math,3.8.
     - Use getline() and stringstream, create a Student, insert as Person*.

Part 7: Input Options – Add Faculty

- **Purpose:** Add faculty to the LinkedList<Person*>.
  4. **From Console:**
     - Prompt for id, name, department, title, salary.
     - Create a Faculty object, insert as Person*.
  2. **From Text File (.txt):**
     - Format: 201 Alice Math Professor 75000.
     - Parse each line, create a Faculty, insert as Person*.
  3. **From CSV File (.csv):**
     - Format: 202,Bob,Science,Assistant Professor,76000.
     - Use getline() and stringstream, create a Faculty, insert as Person*.

Part 8: Sorting Functions

- **Purpose:** Sort the LinkedList<Person*> (add to LinkedList<Person*>).
- **Functions:**
  - **void sortByID():** Sorts by id ascending (applies to both Student and Faculty).
  - **void sortByName():** Sorts alphabetically by name (applies to both).
  - **void sortByGPA():** Sorts students by gpa descending (skips Faculty objects or treats as 0).
- **Logic:** Convert list to a vector of Person*, use std::sort with custom comparators, rebuild the list.

Part 9: Main Menu (Text UI)

Due date: May 12, 11:59 PM

- **Purpose:** A menu to interact with the system using one list.
- **Display:**

```
========== Academic Management System ==========
1. Add Student (Console)
2. Add Students from Text File
3. Add Students from CSV File
4. Add Faculty (Console)
5. Add Faculty from Text File
6. Add Faculty from CSV File
7. Display All Records
8. Search by ID
9. Update by ID
10. Delete by ID
11. Sort by ID
12. Sort by Name
13. Sort by GPA (Students Only)
14. Exit
Enter your choice:
```

Phase 1 File List

- Person.h, Person.cpp
- Student.h, Student.cpp
- Faculty.h, Faculty.cpp
- ListNode.h, ListNode.cpp
- LinkedList.h, LinkedList.cpp
- main.cpp
- Makefile
- p3_report.txt