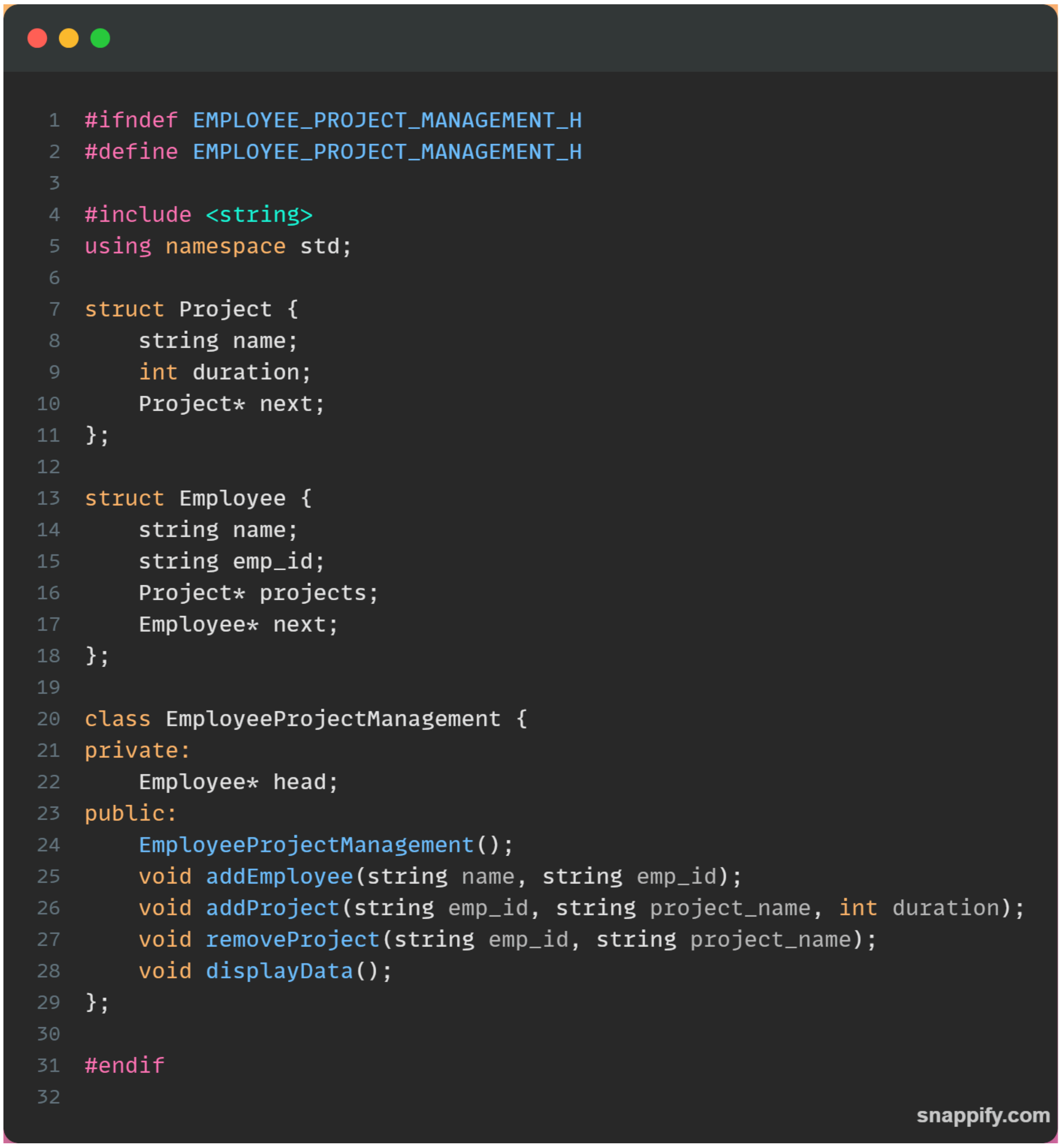
LAPORAN PRAKTIKUM
Modul 13
“Multi Linked List”

Disusun Oleh:
Benedictus Qsota Noventino Baru - 2311104029
S1SE07A

Dosen :
Yudha Islami Sulistya, S.Kom., M.Cs

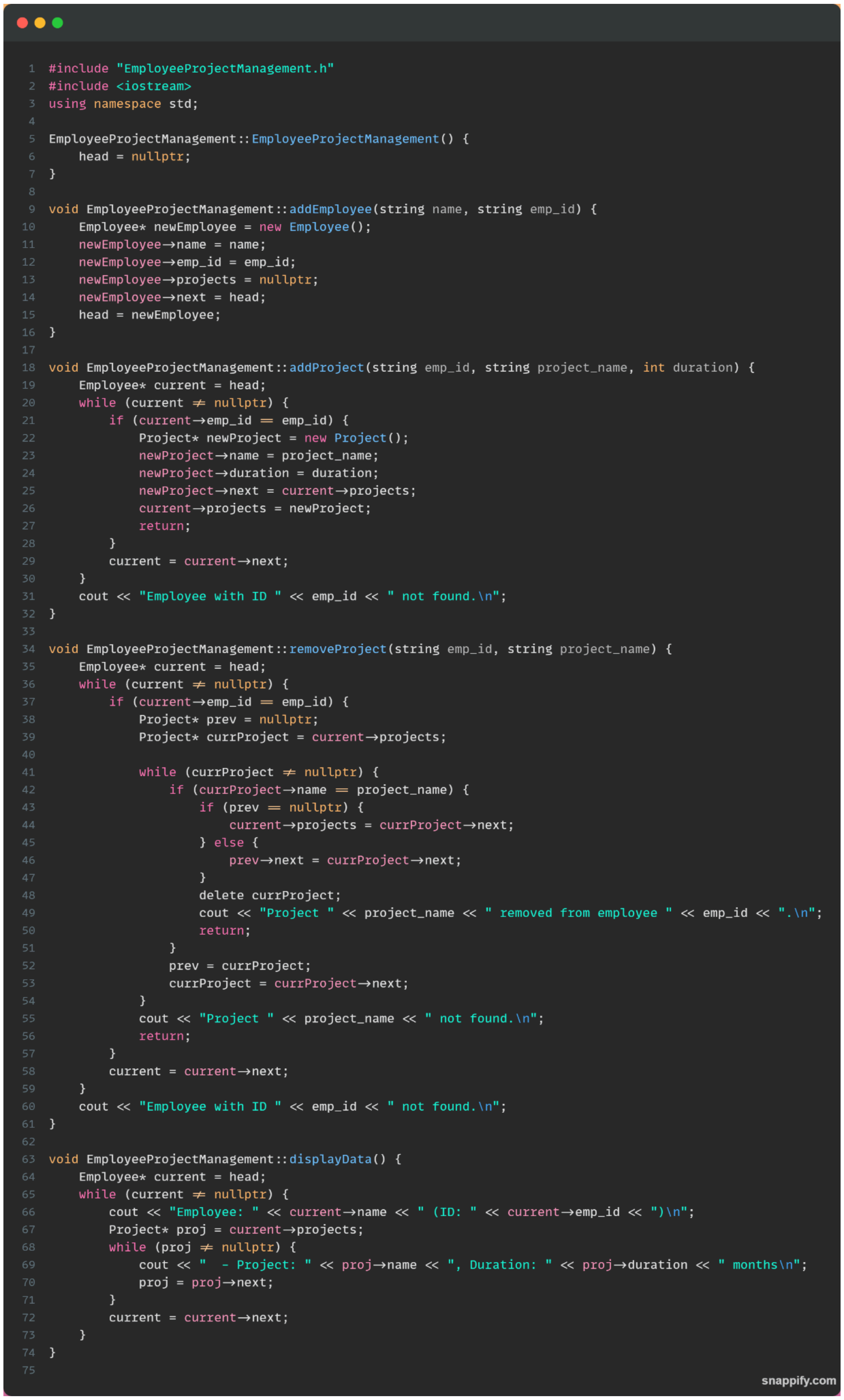
PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM
PURWOKERTO
2024

Latihan Soal Modul
Nomor 1
File EmployeeProjectManagement.h



```cpp
#ifndef EMPLOYEE_PROJECT_MANAGEMENT_H
#define EMPLOYEE_PROJECT_MANAGEMENT_H

#include <string>
using namespace std;

struct Project {
    string name;
    int duration;
    Project* next;
};

struct Employee {
    string name;
    string emp_id;
    Project* projects;
    Employee* next;
};

class EmployeeProjectManagement {
private:
    Employee* head;
public:
    EmployeeProjectManagement();
    void addEmployee(string name, string emp_id);
    void addProject(string emp_id, string project_name, int duration);
    void removeProject(string emp_id, string project_name);
    void displayData();
};

#endif

```

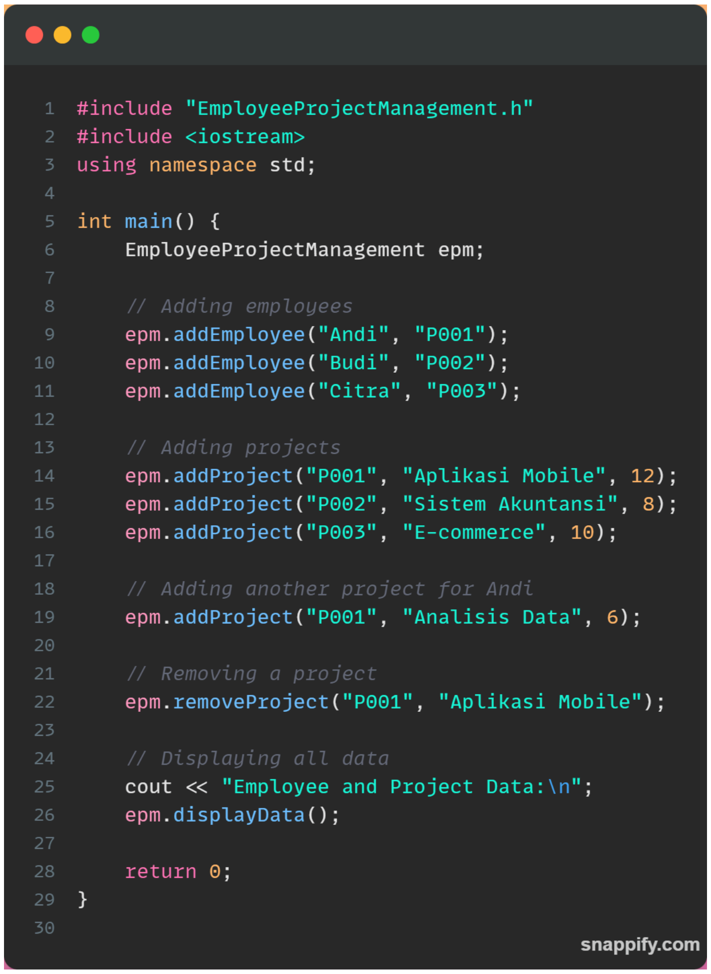
File EmployeeProjectManagement.cpp



```cpp
#include "EmployeeProjectManagement.h"
#include <iostream>
using namespace std;

EmployeeProjectManagement::EmployeeProjectManagement() {
    head = nullptr;
}

void EmployeeProjectManagement::addEmployee(string name, string emp_id) {
    Employee* newEmployee = new Employee();
    newEmployee->name = name;
    newEmployee->emp_id = emp_id;
    newEmployee->projects = nullptr;
    newEmployee->next = head;
    head = newEmployee;
}

void EmployeeProjectManagement::addProject(string emp_id, string project_name, int duration) {
    Employee* current = head;
    while (current != nullptr) {
        if (current->emp_id == emp_id) {
            Project* newProject = new Project();
            newProject->name = project_name;
            newProject->duration = duration;
            newProject->next = current->projects;
            current->projects = newProject;
            return;
        }
        current = current->next;
    }
    cout << "Employee with ID " << emp_id << " not found.\n";
}

void EmployeeProjectManagement::removeProject(string emp_id, string project_name) {
    Employee* current = head;
    while (current != nullptr) {
        if (current->emp_id == emp_id) {
            Project* prev = nullptr;
            Project* currProject = current->projects;

            while (currProject != nullptr) {
                if (currProject->name == project_name) {
                    if (prev == nullptr) {
                        current->projects = currProject->next;
                    } else {
                        prev->next = currProject->next;
                    }
                    delete currProject;
                    cout << "Project " << project_name << " removed from employee " << emp_id << ".\n";
                    return;
                }
                prev = currProject;
                currProject = currProject->next;
            }
            cout << "Project " << project_name << " not found.\n";
            return;
        }
        current = current->next;
    }
    cout << "Employee with ID " << emp_id << " not found.\n";
}

void EmployeeProjectManagement::displayData() {
    Employee* current = head;
    while (current != nullptr) {
        cout << "Employee: " << current->name << " (ID: " << current->emp_id << ")\n";
        Project* proj = current->projects;
        while (proj != nullptr) {
            cout << "  - Project: " << proj->name << ", Duration: " << proj->duration << " months\n";
            proj = proj->next;
        }
        current = current->next;
    }
}

```

snappify.com

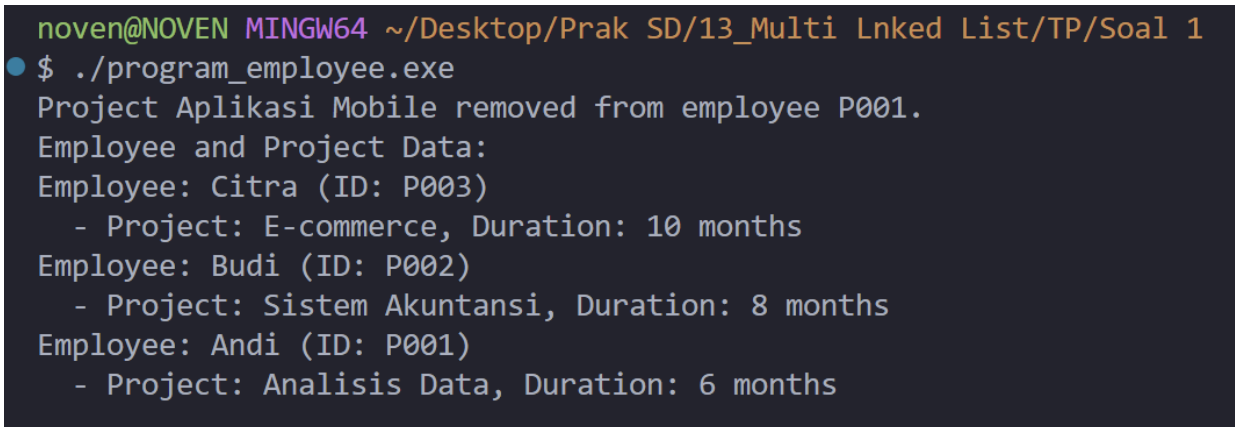
File main.cpp



```cpp
#include "EmployeeProjectManagement.h"
#include <iostream>
using namespace std;

int main() {
    EmployeeProjectManagement epm;

    // Adding employees
    epm.addEmployee("Andi", "P001");
    epm.addEmployee("Budi", "P002");
    epm.addEmployee("Citra", "P003");

    // Adding projects
    epm.addProject("P001", "Aplikasi Mobile", 12);
    epm.addProject("P002", "Sistem Akuntansi", 8);
    epm.addProject("P003", "E-commerce", 10);

    // Adding another project for Andi
    epm.addProject("P001", "Analisis Data", 6);

    // Removing a project
    epm.removeProject("P001", "Aplikasi Mobile");

    // Displaying all data
    cout << "Employee and Project Data:\n";
    epm.displayData();

    return 0;
}

```

snappify.com

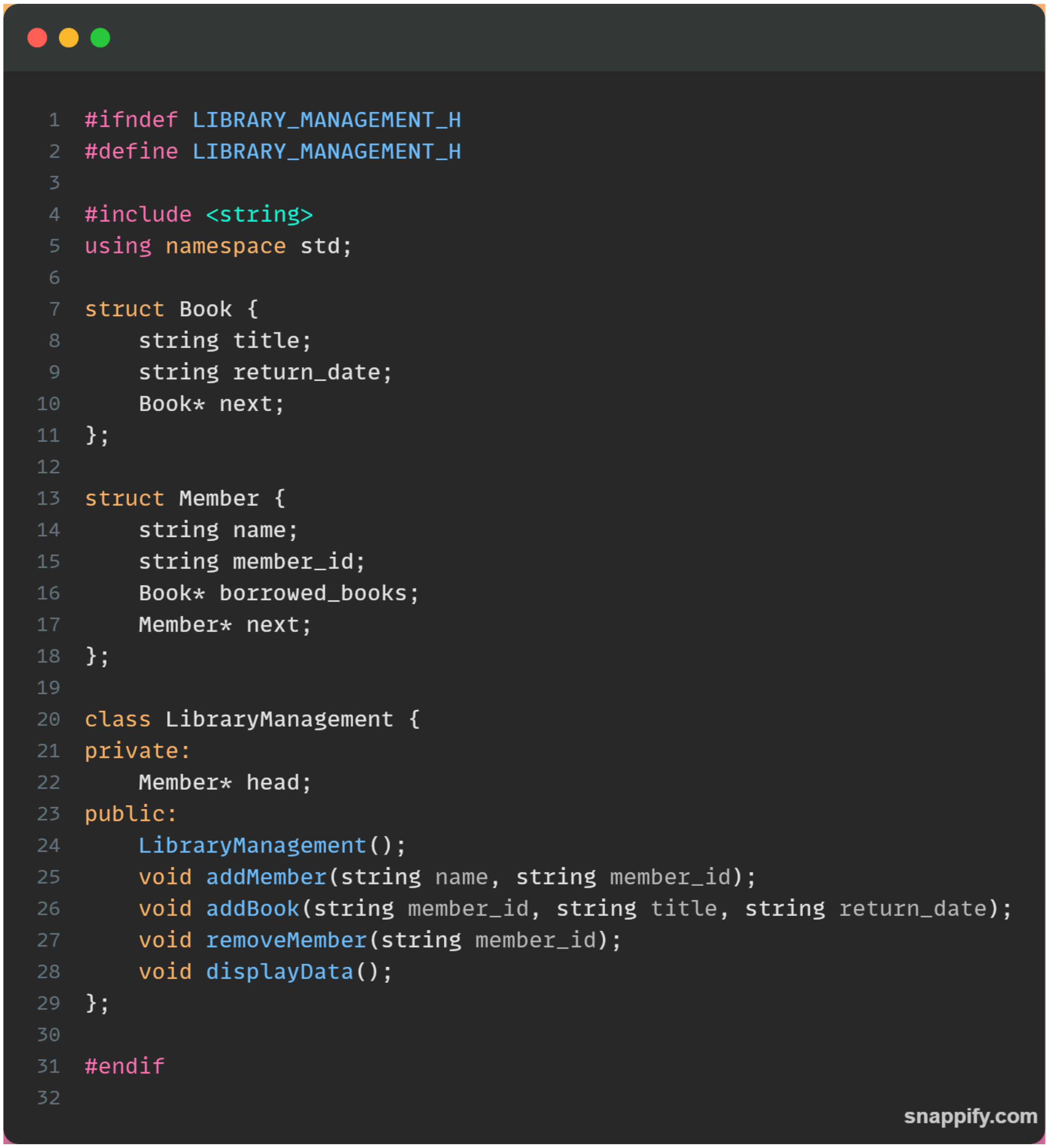
Output



```
noven@NOVEN MINGW64 ~/Desktop/Prak SD/13_Multi Lnked List/TP/Soal 1
$ ./program_employee.exe
Project Aplikasi Mobile removed from employee P001.
Employee and Project Data:
Employee: Citra (ID: P003)
  - Project: E-commerce, Duration: 10 months
Employee: Budi (ID: P002)
  - Project: Sistem Akuntansi, Duration: 8 months
Employee: Andi (ID: P001)
  - Project: Analisis Data, Duration: 6 months
```

Program **Manajemen Data Pegawai dan Proyek** menggunakan **Multi Linked List** untuk mengelola data pegawai dan proyek mereka. Struktur **`Project`** menyimpan nama dan durasi proyek, sedangkan struktur **`Employee`** menyimpan data pegawai beserta daftar proyek yang mereka kelola. Class **`EmployeeProjectManagement`** mengelola operasi seperti menambahkan pegawai, menambahkan proyek ke pegawai tertentu, menghapus proyek, dan menampilkan semua data. Program dimulai dengan menambahkan pegawai seperti **Andi**, **Budi**, dan **Citra**, serta proyek-proyek mereka seperti **Aplikasi Mobile**, **Sistem Akuntansi**, dan **E-commerce**. Proyek baru, seperti **Analisis Data**, juga dapat ditambahkan, dan proyek tertentu, seperti **Aplikasi Mobile**, dapat dihapus dari pegawai yang relevan. Fungsi **`displayData`** menampilkan daftar pegawai beserta proyek yang mereka kelola. Program ini dirancang modular dengan tiga file: header (`EmployeeProjectManagement.h`), implementasi (`EmployeeProjectManagement.cpp`), dan file utama (`program_employee.cpp`).

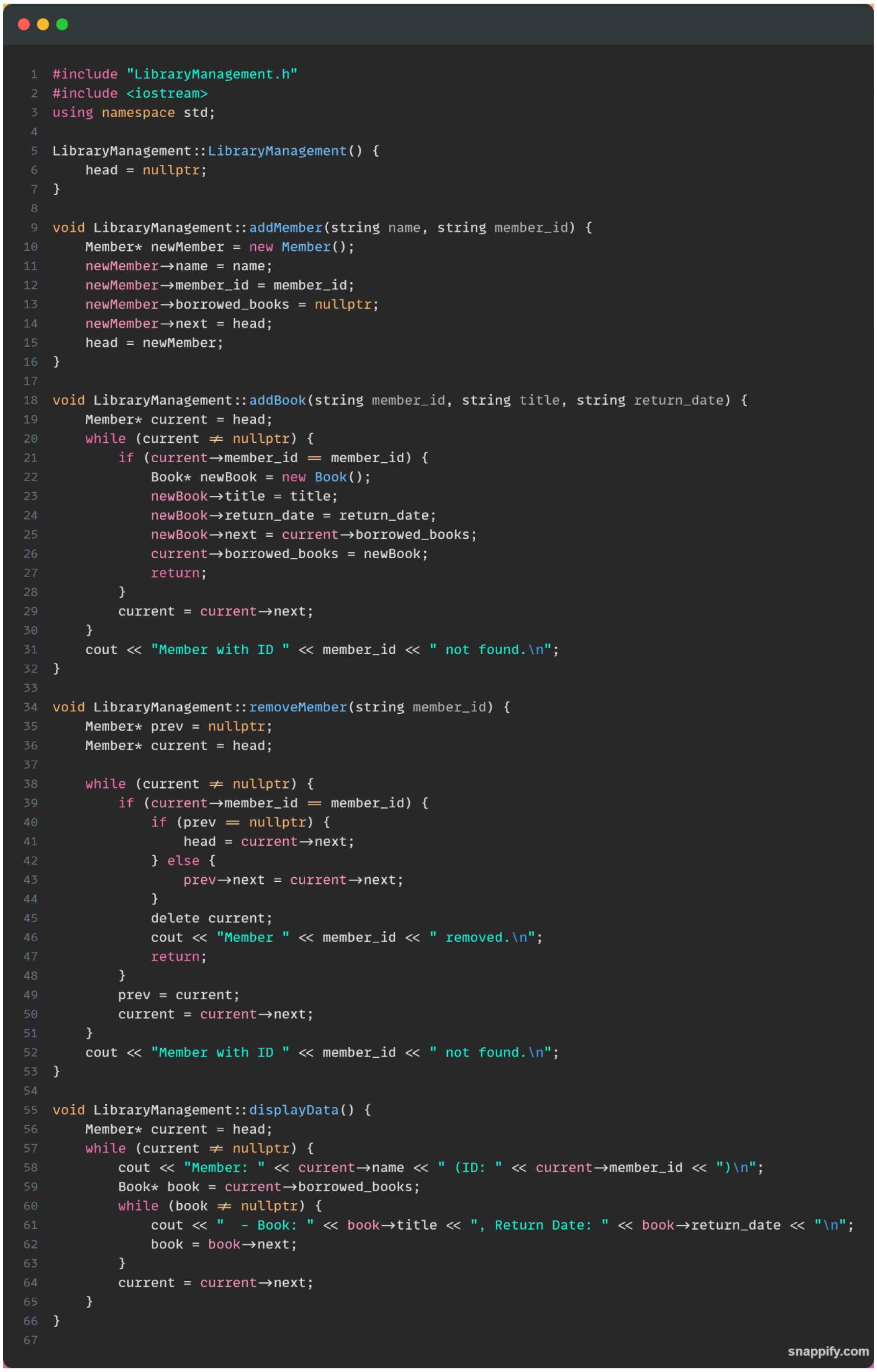
Soal 2
File LibraryManagement.h



```cpp
#ifndef LIBRARY_MANAGEMENT_H
#define LIBRARY_MANAGEMENT_H

#include <string>
using namespace std;

struct Book {
    string title;
    string return_date;
    Book* next;
};

struct Member {
    string name;
    string member_id;
    Book* borrowed_books;
    Member* next;
};

class LibraryManagement {
private:
    Member* head;
public:
    LibraryManagement();
    void addMember(string name, string member_id);
    void addBook(string member_id, string title, string return_date);
    void removeMember(string member_id);
    void displayData();
};

#endif

```

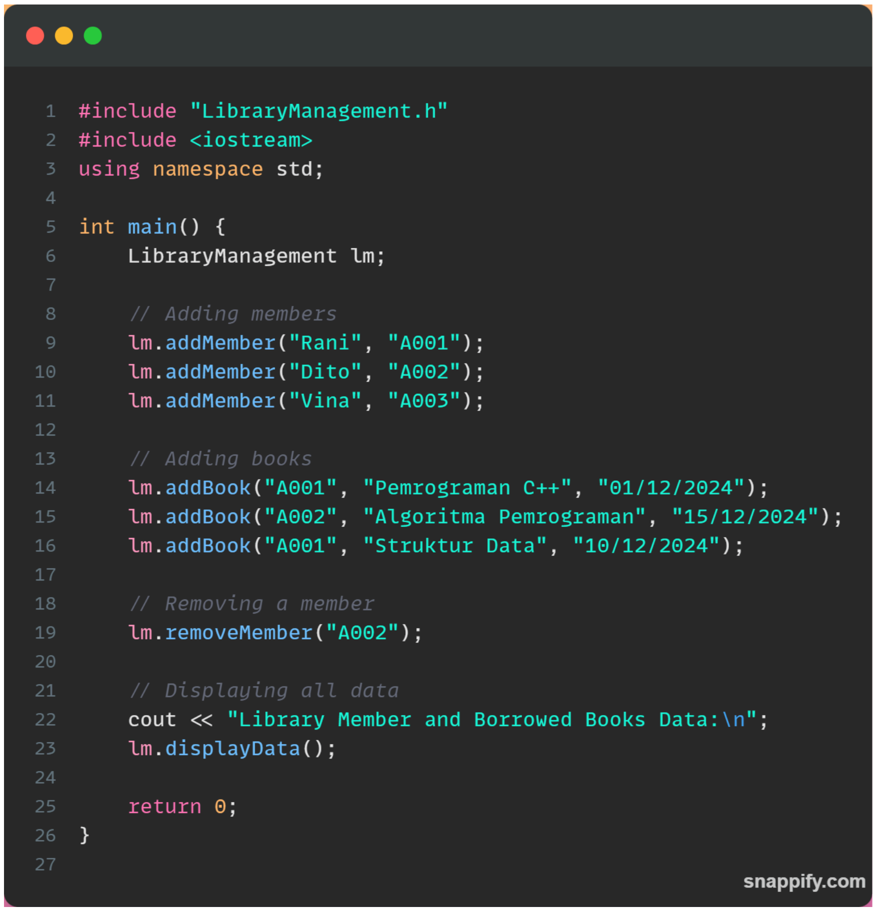
snappify.com

File LibraryManagement.cpp



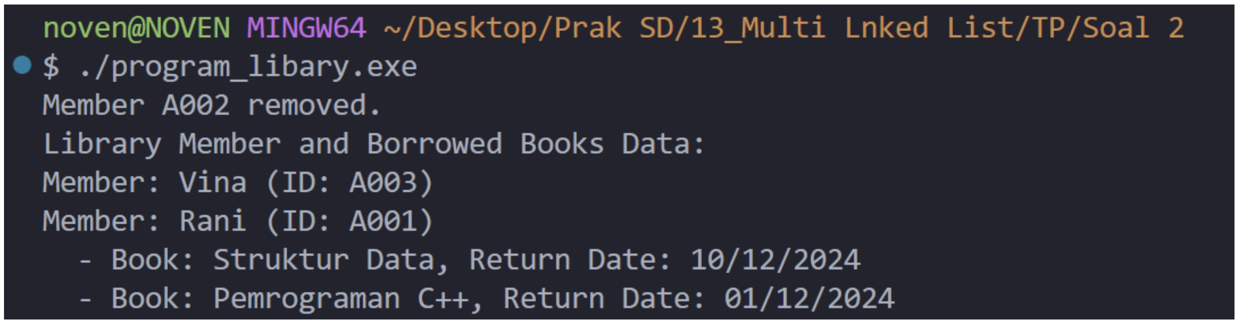
```cpp
#include "LibraryManagement.h"
#include <iostream>
using namespace std;

LibraryManagement::LibraryManagement() {
    head = nullptr;
}

void LibraryManagement::addMember(string name, string member_id) {
    Member* newMember = new Member();
    newMember->name = name;
    newMember->member_id = member_id;
    newMember->borrowed_books = nullptr;
    newMember->next = head;
    head = newMember;
}

void LibraryManagement::addBook(string member_id, string title, string return_date) {
    Member* current = head;
    while (current != nullptr) {
        if (current->member_id == member_id) {
            Book* newBook = new Book();
            newBook->title = title;
            newBook->return_date = return_date;
            newBook->next = current->borrowed_books;
            current->borrowed_books = newBook;
            return;
        }
        current = current->next;
    }
    cout << "Member with ID " << member_id << " not found.\n";
}

void LibraryManagement::removeMember(string member_id) {
    Member* prev = nullptr;
    Member* current = head;

    while (current != nullptr) {
        if (current->member_id == member_id) {
            if (prev == nullptr) {
                head = current->next;
            } else {
                prev->next = current->next;
            }
            delete current;
            cout << "Member " << member_id << " removed.\n";
            return;
        }
        prev = current;
        current = current->next;
    }
    cout << "Member with ID " << member_id << " not found.\n";
}

void LibraryManagement::displayData() {
    Member* current = head;
    while (current != nullptr) {
        cout << "Member: " << current->name << " (ID: " << current->member_id << ")\n";
        Book* book = current->borrowed_books;
        while (book != nullptr) {
            cout << "  - Book: " << book->title << ", Return Date: " << book->return_date << "\n";
            book = book->next;
        }
        current = current->next;
    }
}

```

File main.cpp



```cpp
#include "LibraryManagement.h"
#include <iostream>
using namespace std;

int main() {
    LibraryManagement lm;

    // Adding members
    lm.addMember("Rani", "A001");
    lm.addMember("Dito", "A002");
    lm.addMember("Vina", "A003");

    // Adding books
    lm.addBook("A001", "Pemrograman C++", "01/12/2024");
    lm.addBook("A002", "Algoritma Pemrograman", "15/12/2024");
    lm.addBook("A001", "Struktur Data", "10/12/2024");

    // Removing a member
    lm.removeMember("A002");

    // Displaying all data
    cout << "Library Member and Borrowed Books Data:\n";
    lm.displayData();

    return 0;
}

```

Output



```
noven@NOVEN MINGW64 ~/Desktop/Prak SD/13_Multi Lnked List/TP/Soal 2
$ ./program_libary.exe
Member A002 removed.
Library Member and Borrowed Books Data:
Member: Vina (ID: A003)
Member: Rani (ID: A001)
  - Book: Struktur Data, Return Date: 10/12/2024
  - Book: Pemrograman C++, Return Date: 01/12/2024
```

Program **Sistem Manajemen Buku Perpustakaan** menggunakan **Multi Linked List** untuk mengelola data anggota perpustakaan dan buku yang dipinjam. Struktur **`Book`** menyimpan judul buku dan tanggal pengembalian, sedangkan struktur **`Member`** menyimpan data anggota perpustakaan beserta daftar buku yang dipinjam. Class **`LibraryManagement`** menyediakan fungsi untuk menambahkan anggota, menambahkan buku yang dipinjam, menghapus anggota beserta buku yang dipinjam, dan menampilkan data anggota serta buku yang dipinjam. Program ini dimulai dengan menambahkan anggota seperti **Rani**, **Dito**, dan **Vina**, serta buku yang mereka pinjam, seperti **Pemrograman C++** dan **Algoritma Pemrograman**. Buku baru seperti **Struktur Data** dapat ditambahkan, dan anggota **Dito** dapat dihapus beserta buku yang dipinjam. Fungsi **`displayData`** menampilkan daftar anggota dengan buku yang dipinjam, serta tanggal pengembaliannya. Program ini menggunakan tiga file: header (`LibraryManagement.h`), implementasi (`LibraryManagement.cpp`), dan file utama (`program_liabary.cpp`).