**Lab # 01**

#include <iostream>

#include <vector>

using namespace std;

class Author {

public:

    string name;

    int birthYear;

    Author(string n, int y) : name(n), birthYear(y) {}

};

class Edition {

public:

    string language;

    int pubYear;

    Edition(string lang, int y) : language(lang), pubYear(y) {}

};

class Book {

public:

    string title, ISBN;

    int year;

    Author\* author;

    vector<Edition> editions;

    Book(string t, string i, int y, Author\* a) : title(t), ISBN(i), year(y), author(a) {}

    void addEdition(string lang, int year) {

        editions.push\_back(Edition(lang, year));

    }

};

**Lab # 02**

#include <iostream>

#include <vector>

using namespace std;

class Department;

class Employee {

public:

    int id;

    string name;

    vector<Department\*> departments;

    Employee(int i, string n) : id(i), name(n) {}

};

class Department {

public:

    string name, location;

    Employee\* manager;

    Department(string n, string loc, Employee\* m) : name(n), location(loc), manager(m) {}

};

**Lab # 03**

#include <iostream>

#include <vector>

using namespace std;

class Cylinder

{

public:

    float diameter;

    string material;

    Cylinder(float d, string m) : diameter(d), material(m) {}

};

class Engine

{

public:

    string engineNumber;

    int horsePower;

    vector<Cylinder> cylinders;

    Engine(string num, int hp) : engineNumber(num), horsePower(hp) {}

};

class Tire

{

public:

    string brand;

    float treadDepth;

    Tire(string b, float td) : brand(b), treadDepth(td) {}

};

class Wheel

{

public:

    int size;

    string type;

    Tire tire;

    Wheel(int s, string t, Tire ti) : size(s), type(t), tire(ti) {}

};

class Seat

{

public:

    string material;

    bool isHeated;

    Seat(string m, bool h) : material(m), isHeated(h) {}

};

class Car

{

public:

    string model, make;

    int year;

    Engine engine;

    vector<Wheel> wheels;

    vector<Seat> seats;

    Car(string mo, string ma, int y, Engine e) : model(mo), make(ma), year(y), engine(e) {}

};

**Lab # 04**

#include <iostream>

#include <vector>

using namespace std;

class Page

{

public:

    int number;

    string content;

    Page(int n, string c) : number(n), content(c) {}

};

class Chapter

{

public:

    int number;

    string title, summary;

    vector<Page> pages;

    Chapter(int n, string t, string s) : number(n), title(t), summary(s) {}

    void addPage(int n, string c)

    {

        pages.emplace\_back(n, c);

    }

};

class Book

{

public:

    string title, ISBN, author;

    vector<Chapter> chapters;

    Book(string t, string i, string a) : title(t), ISBN(i), author(a) {}

    void addChapter(Chapter c)

    {

        chapters.push\_back(c);

    }

};

**Lab # 05**

#include <iostream>

#include <vector>

using namespace std;

class Account

{

public:

    int accNumber;

    float balance;

    Account(int num) : accNumber(num), balance(0) {}

    virtual void deposit(float amt) { balance += amt; }

    virtual void withdraw(float amt) { balance -= amt; }

    virtual void statement() const

    {

        cout << "Acc#: " << accNumber << ", Balance: " << balance << "\n";

    }

};

class SavingsAccount : public Account

{

public:

    SavingsAccount(int num) : Account(num) {}

};

class CurrentAccount : public Account

{

public:

    CurrentAccount(int num) : Account(num) {}

};

class Customer

{

public:

    int id;

    string name;

    vector<Account \*> accounts;

    Customer(int i, string n) : id(i), name(n) {}

    void addAccount(Account \*a) { accounts.push\_back(a); }

};

**Lab # 06**

#include <iostream>

#include <vector>

using namespace std;

class Person

{

public:

    int id;

    string name;

    Person(int i, string n) : id(i), name(n) {}

};

class Course

{

public:

    string code, title;

    Person \*teacher;

    Course(string c, string t, Person \*teach) : code(c), title(t), teacher(teach) {}

};

class Transcript

{

public:

    Course \*course;

    string grade;

    Transcript(Course \*c, string g) : course(c), grade(g) {}

};

class Student : public Person

{

public:

    vector<Course \*> enrolled;

    vector<Transcript> transcript;

    Student(int i, string n) : Person(i, n) {}

    void enroll(Course \*c) { enrolled.push\_back(c); }

    void addGrade(Course \*c, string g) { transcript.emplace\_back(c, g); }

};

class Teacher : public Person

{

public:

    string dept;

    vector<Course \*> teaches;

    Teacher(int i, string n, string d) : Person(i, n), dept(d) {}

};

class Department

{

public:

    string name;

    vector<Teacher \*> teachers;

    vector<Course \*> courses;

};

class Appointment;

class Patient

{

public:

    string name, history;

    vector<Appointment \*> appointments;

};

class Doctor

{

public:

    string name, department;

};

class Nurse

{

public:

    string name;

    vector<Patient \*> patients;

    vector<Doctor \*> doctors;

};

class Appointment

{

public:

    Patient \*patient;

    Doctor \*doctor;

};