Problem 1

In boolean algebra, A ^ (B U C) = (A ^ B) U (A ^ C). Your task is to show that this law holds by checking the equality for all combinations of A, B, and C values. To assist you in this task, there are two classes that implement the functionality of AND and OR gates, and both inherit from a common class called Gate. The classes have two protected attributes for boolean inputs (which are set by the constructor) and a method to get the output.

```
#include <iostream>
using namespace std;

class Gate {
protected:
    ???

public:
    Gate(bool in1, bool in2) : input1(in1), input2(in2) {}

    virtual bool getOutput() const = 0;

    virtual ~Gate() {}
};

class AndGate : public Gate {
public:
    AndGate(bool in1, bool in2) : Gate(in1, in2) {}

    bool getOutput() const override {
        return input1 && input2;
    }
};

class OrGate : public Gate {
public:
    ???
};

int main() {
    for (bool a : {false, true}) {
        for (bool b : {false, true}) {
            for (bool c : {false, true}) {
                cout << "A: " << a << "; B: " << b << "; C: " << c << '\n';

                OrGate o1(b, c);
                AndGate a1(a, o1.getOutput());
                cout << "A ^ (B U C): " << a1.getOutput() << '\n';

                ???
                cout << "(A ^ B) U (A ^ C): " << ??? << "\n\n";
            }
        }
    }

    return 0;
}
```

Next task: Edit the program to prove the other distributive law of boolean algebra, which is A U (B ^ C) = (A U B) ^ (A U C).

Problem 2

A shift cipher is an encrypting scheme where in a message is transformed by shifting every letter in the input string by a fixed displacement. For example, 'A' would become 'D' when a shift value of 3 is used. Write a program with two classes for encryption and decryption that inherit from a common Transformer class. The classes must have two protected attributes — the input string, and an integer holding shift value. When transforming a string, non-alphabet characters are left as they are. For example, 'Hello World!' shifted by 5 would become 'Mjqqt Btwqi!'.

```
#include <iostream>
#include <string>
using namespace std;

class Transformer {
protected:
    ???

public:
          Transformer(int  shift,  const  string&  input)  :  shiftValue(shift),
inputString(input) {}

    virtual string transform() const = 0;

    virtual ~Transformer() {}
};

class Encrypter : public Transformer {
public:
    Encrypter(int shift, const string& input) : Transformer(shift, input) {}

    string transform() const override {
        ???
        return ???;
    }
};

class Decrypter : public Transformer {
public:
    ???
};

int main() {
    string input;
    int shift;

    cout << "Enter a string to encrypt: ";
    getline(cin, input);
    cout << "Enter shift value: ";
    cin >> shift;

    ???
    cout << "Encrypted string: " << ??? << endl;

    Decrypter decrypter(shift, encrypted);
    string decrypted = decrypter.transform();
    cout << "Decrypted string: " << decrypted << endl;

    return 0;
}
```

Next task: Edit the program to implement the Vigenere cipher. The Transformer class must have a string attribute called key instead of the integer displacement.

Problem 3

The university that you work at has 3 kinds of labs — software, hardware, and hybrid. You have to write a program that should handle the allocation of labs. Your program must have a template class for managing a list of labs, and you must use it to create 3 lists — one list for each kind of lab.

Your program must handle queries. Each query begins with an integer x, which determines the type of query.

1. If x = 1, you must free a lab. You will have to input a character c which indicates the type of lab (S = software, H = hardware, and X = hybrid), and another integer y, which indicates the index of the lab to be freed. For example, input '1 X 3' means you must free the 3rd hybrid lab.
2. If x = 2, you must find a free lab, occupy it, and display its information. You will be given a character c indicating the type of lab that is requested (S = software, H = hardware). Note that a hybrid lab can be used to satisfy either query. If no lab is available, print it to the user. You must implement the code for this type of query from scratch. An example of this query is input '2 S', which means you must find a free software or hybrid lab for the user.

```
#include <iostream>
#include <string>
#include <iomanip>
using namespace std;

const size_t MAX_LABS = 10;

class Lab {
protected:
    string name;
    bool isOccupied;

public:
    Lab() : name(""), isOccupied(false) {}
    Lab(const string& name, bool occupied)
        : name(name), isOccupied(occupied) {}

    void freeLab() { isOccupied = false; }

    void occupyLab() { isOccupied = true; }

    bool getOccupiedStatus() const { return isOccupied; }

    virtual void displayInfo() const = 0;

    virtual ~Lab() {}
};

class SWLab : public Lab {
private:
    int numComputers;

public:
    SWLab() : Lab(), numComputers(30) {}
    SWLab(const string& name, bool occupied)
        : Lab(name, occupied), numComputers(30) {}

    void displayInfo() const override {
```

```
             cout << "Software Lab - Name: " << name << ", Computers: " <<
numComputers
             << ", Occupied: " << (isOccupied ? "Yes" : "No") << endl;
    }
};

class HWLab : public Lab {
private:
    int numDevices;

public:
    HWLab() : Lab(), numDevices(40) {}
    HWLab(const string& name, bool occupied)
        : Lab(name, occupied), numDevices(40) {}

    void displayInfo() const override {
        cout << "Hardware Lab - Name: " << name << ", Devices: " << numDevices
             << ", Occupied: " << (isOccupied ? "Yes" : "No") << endl;
    }
};

class HybridLab : public Lab {
private:
    int numComputers;
    int numDevices;

public:
    HybridLab() : Lab(), numComputers(10), numDevices(20) {}
    HybridLab(const string& name, bool occupied)
        : Lab(name, occupied), numComputers(10), numDevices(20) {}

    void displayInfo() const override {
        cout << "Hybrid Lab - Name: " << name << ", Computers: " << numComputers
             << ", Devices: " << numDevices
             << ", Occupied: " << (isOccupied ? "Yes" : "No") << endl;
    }
};

template <typename T>
class LabList {
private:
    T labs[MAX_LABS];
    size_t count;

public:
    LabList() : count(MAX_LABS) {}

    void setup(const string& labType) {
        for (size_t i = 0; i < MAX_LABS; ++i) {
            labs[i] = T(labType + "-Lab-" + to_string(i + 1), false);
        }
    }

    void freeLab(size_t index) {
        labs[index].freeLab();
    }

    size_t getFreeLabCount() const {
        ???
    }

    T* occupyFreeLab() {
        for (size_t i = 0; i < MAX_LABS; ++i) {
            if (!labs[i].getOccupiedStatus()) {
                labs[i].occupyLab();
                return &labs[i];
```

```
            }
        }
        return nullptr;
    }

    void displayAll() const {
        for (size_t i = 0; i < MAX_LABS; ++i) {
            labs[i].displayInfo();
        }
    }
};

int main() {
    LabList<SWLab> swLabs;
    swLabs.setup("Software");
    LabList<HWLab> hwLabs;
    hwLabs.setup("Hardware");
    LabList<HybridLab> hybridLabs;
    hybridLabs.setup("Hybrid");

    while (true) {
        int x;
        cin >> x;
        if (x == 1) {
            char c;
            int y;
            cin >> c >> y;
            if (c == 'S') swLabs.freeLab(y - 1);
            if (c == 'H') ???
            if (c == 'X') hybridLabs.freeLab(y - 1);
        } else if (x == 2) {
            ???
        }
    }

    return 0;
}
```

Next task: Add a third type of query, denoted by x = 3. This query requires you to print the info of all free labs of one type. For example, input ‘3 X’ means you must print the info of all free hybrid labs. Note that adding a method to the template class might help you here.

Problem 4

Write a program that calculates flight times between cities. There must be a class City whose friend class is Flight, allowing it to access information about the city such as code (which is a unique identifier) and coordinates. Flight has a speed attribute which is used to calculate flight times.

```
#include <iostream>
#include <vector>
#include <cmath>
#include <string>
using namespace std;

class Flight;

class City {
private:
    ???

public:
    City(string c, string n, double xCoord, double yCoord)
        : code(c), name(n), x(xCoord), y(yCoord) {}

    friend class Flight;
};

class Flight {
private:
    double speed;
    vector<City> cities;

public:
    Flight(double s) : speed(s) {}

    void addCity(const City& city) {
        cities.push_back(city);
    }

    City* getCityByCode(const string& code) {
        for (auto& city : cities) {
            if (city.code == code) {
                return &city;
            }
        }
        return nullptr;
    }

    void booking(const string& code1, const string& code2) {
        City* city1 = getCityByCode(code1);
        City* city2 = getCityByCode(code2);

        ???

        cout << "Flight time from " << city1->name << " to " << city2->name
        << ": " << flightTime << " hours" << endl;
    }
};

int main() {
    City city1("NY", "New York", 40.7128, -74.0060);
    City city2("LA", "Los Angeles", 34.0522, -118.2437);
    City city3("SF", "San Francisco", 37.7749, -122.4194);

    Flight flight(7.5);
```

```
    flight.addCity(city1);
    flight.addCity(city2);
    flight.addCity(city3);

    flight.booking("NY", "LA");
    flight.booking("LA", "SF");
    flight.booking("NY", "SF");

    return 0;
}
```

Next task: Edit Flight to keep track of earnings. Flight must have two additional attributes — rate, which is the price per unit distance, and earnings, which is the money made till now. Every time the booking method is called, calculate the price of that particular flight, print it, and add it to earnings. Add a method to Flight that displays current earnings.

Problem 5

Your task is to write a Virus class that has a method that corrupts an attribute string in a certain way. You must implement SwapVirus, which is a child of Virus and corrupts the input string by randomly picking two indices and swapping the characters at them. For random number generation, you must write a pseudo-random number generator, that uses linear congruential generation. Linear congruential generation works on the formula X' = (A * X + C) % M, where X is the current number, X' is the next number, and A, C, and M are fixed attributes of the generator. Of course, the generator has to be initialised with a number X at the start, for which we will be using the current time (this is common practice for pseudo random number generators).

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <ctime>

using namespace std;

class PRNG {
private:
    int a, c, m;
    int x;

public:
    PRNG(int _a, int _c, int _m, int seed = 0) : a(_a), c(_c), m(_m), x(seed) {}

    int next() {
        x = ???;
        return x;
    }
};

class Virus {
protected:
    string data;
    PRNG prng;

public:
    Virus(const string& _data, const PRNG& _prng) : data(_data), prng(_prng) {}

    virtual void corrupt() = 0;

    void display() const {
        ???
    }
};

class SwapVirus : public Virus {
public:
    SwapVirus(const string& _data, const PRNG& _prng) : Virus(_data, _prng) {}

    void corrupt() override {
        int index1 = prng.next() % data.length();
        int index2 = prng.next() % data.length();
        ???
    }
};

int main() {
    int a = 1664525, c = 1013904223, m = 1024;
    PRNG prng(a, c, m, static_cast<int>(time(0)));
```

```
    string virusData = "Hello World!";å
    SwapVirus swapVirus(virusData, prng);
    swapVirus.display();

    for (int i = 0; i < 10; i++) {
        swapVirus.corrupt();
        swapVirus.display();
    }

    return 0;
}
```

Next task: You must write a new child class of Virus called ASCIIVirus. ASCIIVirus picks a random index and places a random ASCII character at that index.