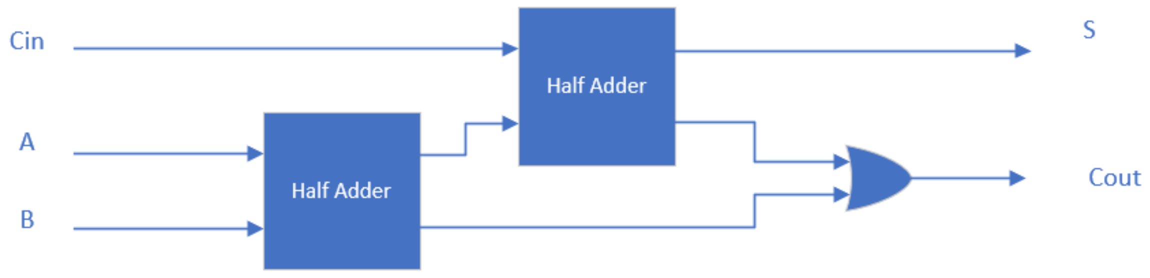
P5 – Digital Objects

Prep

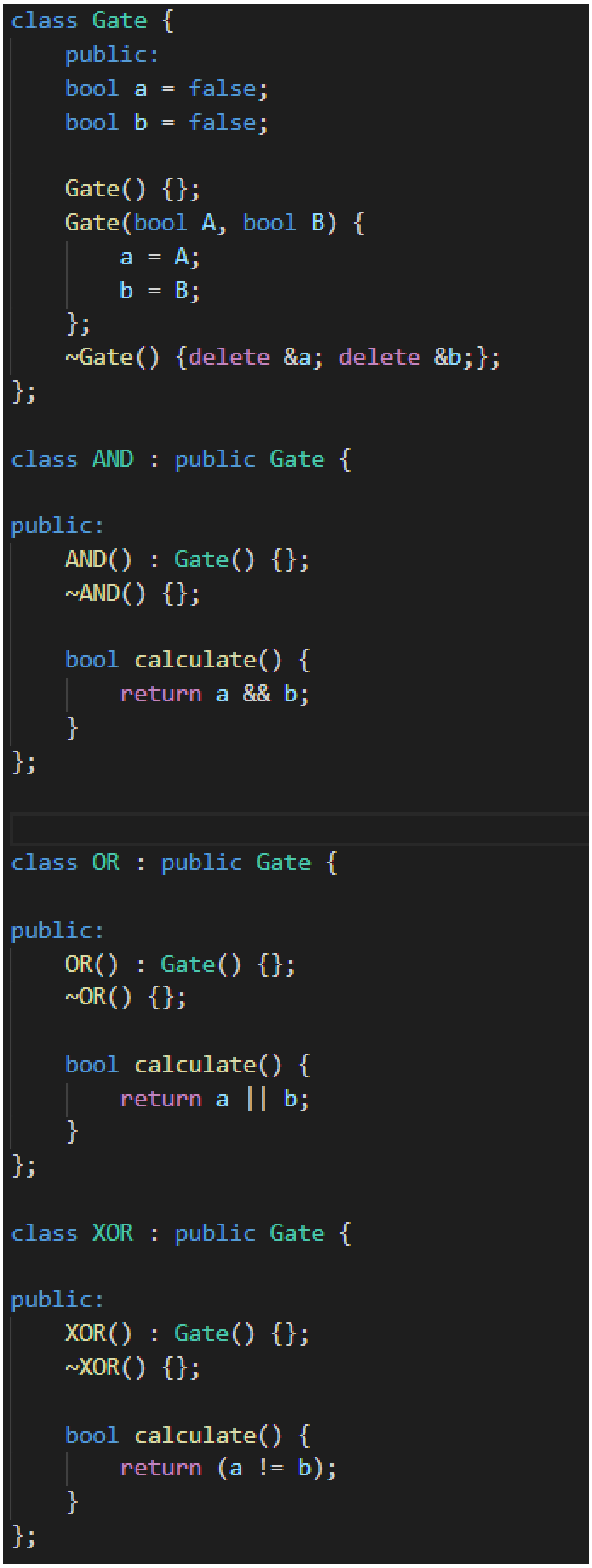
In order to implement my logic gate adder using objects I will need to use inheritance to make it easier. For my gates I will have a Gate class which will handle the input and output of the logic, and then I will have individual classes for all the logical operators I need, for example AND and XOR, which will inherit this Gate class.

Firstly, a half adder simply takes in two bits (a and b) and then XORs them to produce the sum and ANDs them to produce the carry. Therefore, my HalfAdder object must inherit an XOR gate and an AND gate.

Secondly as described by my diagram below, a full adder can be made from two half adders and an OR gate. Therefore, my FullAdder class can therefore inherit two HalfAdder classes and one OR gate class.



The delete and delete [] expressions should be used in the destructor of an object to free up the memory after the object is deleted. This means that no memory will be taken up after the object has been deleted.



```cpp
class Gate {
    public:
    bool a = false;
    bool b = false;

    Gate() {};
    Gate(bool A, bool B) {
        a = A;
        b = B;
    };
    ~Gate() {delete &a; delete &b;};
};

class AND : public Gate {

public:
    AND() : Gate() {};
    ~AND() {};

    bool calculate() {
        return a && b;
    }
};


class OR : public Gate {

public:
    OR() : Gate() {};
    ~OR() {};

    bool calculate() {
        return a || b;
    }
};

class XOR : public Gate {

public:
    XOR() : Gate() {};
    ~XOR() {};

    bool calculate() {
        return (a != b);
    }
};
```