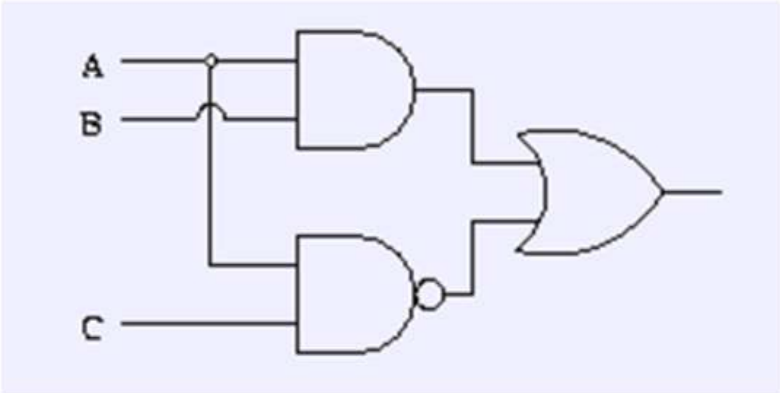
Design a simple logic simulator that provides a mechanism to run and simulate logic circuits then find the desired outputs for different designs.

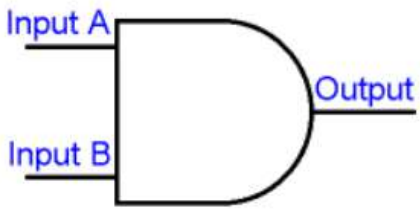


Create a class called "Node" that has name and value data members to represent each node in the circuit such as node A, B in the above circuit.

Class "Node" should have the following Specifications:

- Default and non-default Constructor.
- Provide setters and getters.
- Overload the operator ostream << to print the node information.
- Create methods called AND() / OR() / XOR() to perform the logical operation between two nodes objects.
- Add any needed functions according to your design.

Create a class Called "Gate" which has two input nodes and one output node data members.



Class Gate should support the following methods:

- Default and non-default Constructors.
- Provide getters and setters for its nodes.
- Implement the gate types AND / OR / NAND / NOR / XOR / XNOR / NOT.
- function simulateGate() returns the logic value of the gate according to its type.

Create a class called "Simulator" which accpets all types of gates then calculate circuits' outputs, It has the following specifications:

- array of pointers to Gate and array of pointers to Node.
- Method postGate() accepts a pointer to a created gate to insert it into the array.
- Method postNode() accepts a pointer to a created node to insert it into the array.
- Method FindNode() accepts a string node's name to find it in the container and return its address.
- Method startSimulate() start the simulation for each gate by looping over the gates container

Note: Only one object can be created from the Simualtor class.

Recommended: Make the gates container and nodes container as a vector to be easliy dynamic allocated during the simulation.

Vector Documentation: https://www.programiz.com/cpp-programming/vectors

Create a class called "GateGenerator" that generates the nodes and all sepcified gates from the input then **post** them to the **simulator**.

"GateGenerator" class should have the following specifications:

- Function parseInput() reads the inputs and parses each keyword ("SIM"/"OUT",..) to the appropriate logic.
- Function createNode() creates a node and return its address.
- Function createGate() creates a (AND,OR,...) gates according to the input and return its address, it is advisable to use a factory function to have better practice on polymorphic objects.
- Add any needed functions as your design says so!.

Input Format

The input follows these formats:

1. gate type (such as AND/OR gate)
2. The inputs and output symbols
3. Write "SET" values to the inputs (either 0 or 1)
4. Write "SIM" to start simulation
5. Write "OUT" to view the output of a specific node or "OUT ALL" to view all nodes' values

Constraints

- Do not use C type casting.
- Do not use void pointers.
- Do not use friend functions or friend classes except (ostream overloading function).
- Do not declare any variable public.
- Do not declare all your functions public. Make decisions for which ones will be public, private or protected.
- The code should have 0 warning.

Output Format

Output format should be the node symbol: node Value.

Sample Input 1

```
AND A B D
NAND A C E
OR D E F
SET A 0
SET B 1
SET C 1
SIM
OUT F
OUT ALL
```

Sample Output 1

```
F: 1
A: 0
B: 1
D: 0
C: 1
E: 1
F: 1
```