```
// TestSignalsChain.cpp
//
// Implementing a simple Chain of Responsibility pattern using Boost
// signal2. (3-layer hardware/software model).
//
// Related also to Observer pattern.
//
// Create a signal with a given signature and attach
// signature-compatible slots to it.
//
// Signals are signature-based.
//(DJD)
//
// (C) Datasim Education BV 2008-2023
//

// DEPRECATED #include <boost/signals.hpp>
#include <boost/signals2/signal.hpp>
#include <iostream>

void hw(double& d)
{
	if (d < 2.0 || d > 4.0)
	{
		std::cout << "Data truncate in hw layer\n";
		d = 3.0;
	}
	std::cout << "the hardware layer, value is: " << d << '\n';
}

void data(double& d)
{
	d *= 2.0;
	std::cout << "the data layer, value: " << d <<  '\n';
}

void comm(double& d)
{
	std::cout << "the communication layer, value: " << d <<  '\n';
}

void Agent()
{ // The mediator

	double d = 1.0;
	hw(d);
	data(d);
	comm(d);
}

int main()
{

//	Agent();

	// HW->Data->Comm
	std::cout << "Chain reaction\n";
	boost::signals2::signal<void (double& d)> signalExterior;
```

```
        boost::signals2::signal<void (double& d)> signalHW;
        boost::signals2::signal<void (double& d)> signalData;
        boost::signals2::signal<void (double& d)> signalComm;

        // Build the connections
        signalHW.connect(&hw);
        signalData.connect(&data);
        signalComm.connect(&comm);

        signalExterior.connect(signalHW);
        signalHW.connect(signalData);
        signalData.connect(signalComm);

//      boost::signals2::signal<void(double& d)> signalExterior;
        double value = -3.7; // in range [2.0, 5.0]
        signalExterior(value);

        return 0;
}
```