```
/* The following code example is taken from the book
 * "The C++ Standard Library - A Tutorial and Reference, 2nd Edition"
 * by Nicolai M. Josuttis, Addison-Wesley, 2012
 *
 * (C) Copyright Nicolai M. Josuttis 2012.
 * Permission to copy, use, modify, sell and distribute this software
 * is granted provided this copyright notice appears in all copies.
 * This software is provided "as is" without express or implied
 * warranty, and with no claim as to its suitability for any purpose.
 */
#include <condition_variable>
#include <mutex>
#include <future>
#include <thread>
#include <iostream>
#include <queue>

std::queue<int> queue;
std::mutex queueMutex;
std::condition_variable queueCondVar;

void provider (int val)
{
    // push different values (val til val+5 with timeouts of val milliseconds
into the queue
    for (int i=0; i<6; ++i) {
        {
            std::lock_guard<std::mutex> lg(queueMutex);
            queue.push(val+i);
        } // release lock
        queueCondVar.notify_one();

        std::this_thread::sleep_for(std::chrono::milliseconds(val));
    }
}

void consumer (int num)
{
    // pop values if available (num identifies the consumer)
    while (true) {
        int val;
        {
            std::unique_lock<std::mutex> ul(queueMutex);
            queueCondVar.wait(ul,[]{ return !queue.empty(); });
            val = queue.front();
            queue.pop();
        } // release lock
        std::cout << "consumer " << num << ": " << val << std::endl;
    }
}

int main()
{
    // start three providers for values 100+, 300+, and 500+
    auto p1 = std::async(std::launch::async,provider,100);
    auto p2 = std::async(std::launch::async,provider,300);
```

```
    auto p3 = std::async(std::launch::async,provider,500);

    // start two consumers printing the values
    auto c1 = std::async(std::launch::async,consumer,1);
    auto c2 = std::async(std::launch::async,consumer,2);
}
```