<https://www.reddit.com/r/AskProgramming/comments/r7607e/c_multithreading_sleeping_barber_problem_help/>

Fixes to this code:

1. No one wakes the barber:
2. And how do customers get haircuts:

Holds issues:::

// Example program

#include <iostream>

#include <string>

#include <mutex>

#include <thread>

#include <chrono>

#include <atomic>

#include <deque>

#include <vector>

#include <condition\_variable>

using namespace std;

const unsigned int NUM\_CHAIRS = 2;

const unsigned int NUM\_CUSTOMERS = 1;

// In a barbershop, customers are coming in and there is only one barber. If the barber is cutting a

// customers hair, the customer will wait. There are only a limited amount of chairs for the customer

// to wait on. If the chairs are full, then the customer will leave and try again later.

struct Barbershop

{

public:

Barbershop(): num\_waiting(0){};

condition\_variable cv; // Acts like a receptionist

atomic<int> num\_waiting; // Number of customers waiting on chairs

mutex cout\_mtx;

};

class Barber

{

public:

Barber(Barbershop& shop): mem\_shop(shop), hislife(&Barber::work, this){};

void work()

{

unique\_lock<mutex> lk(b\_mu);

if(mem\_shop.num\_waiting == 0)

{

mem\_shop.cout\_mtx.lock();

cout << "Barber sleeping...\n";

mem\_shop.cout\_mtx.unlock();

}

mem\_shop.cv.wait(lk, [this]{return mem\_shop.num\_waiting > 0;}); // []{return mem\_shop.num\_waiting}

mem\_shop.cout\_mtx.lock();

cout << "Time to cut some hair!\n";

mem\_shop.cout\_mtx.unlock();

lk.unlock();

mem\_shop.cv.notify\_one();

this\_thread::sleep\_for(chrono::milliseconds(100));

mem\_shop.num\_waiting--;

};

~Barber()

{

hislife.join();

}

int jn;

mutex b\_mu;

Barbershop& mem\_shop;

thread hislife;

};

class Customer

{

public:

Customer(Barbershop& shop, int name\_id): mem\_shop(shop), name(name\_id), hislife(&Customer::tryGetHaircut, this){};

void tryGetHaircut()

{

if(mem\_shop.num\_waiting == NUM\_CHAIRS)

{

mem\_shop.cout\_mtx.lock();

cout << name <<"'s wait was too long. Leaving!\n";

mem\_shop.cout\_mtx.unlock();

}

else

{

mem\_shop.num\_waiting++;

std::unique\_lock<mutex> lk(c\_mu);

mem\_shop.cv.wait(lk);

mem\_shop.cout\_mtx.lock();

cout << name << " is getting haircut!\n";

mem\_shop.cout\_mtx.unlock();

}

};

~Customer()

{

hislife.join();

}

mutex c\_mu;

Barbershop& mem\_shop;

thread hislife;

unsigned int name;

};

int main()

{

Barbershop mybarbershop;

Barber dan();

vector<Customer\*> customers;

customers.reserve(NUM\_CUSTOMERS-1);

for(unsigned int i=0; i<NUM\_CUSTOMERS; i++)

{

customers.push\_back(new Customer(mybarbershop, i+1));

}

for(auto s : customers)

{

delete s;

}

cout << "Program Finished!\n";

return 0;

}