```
#include <string>
#include <forward_list>
#include <iostream>


class Machine {
private:
  std::shared_ptr<MachineState> _state = std::make_shared<OpenOffState>( std::make_shared<Machine>());
public:
  void setState(std::shared_ptr<MachineState> state) { _state = state; }

  void power()  { _state->power();}
  void open()   { _state->open(); }
  void tick()   { _state->tick(); }
  void status() { std::cout << _state->status(); }
};
class MachineState{
protected:
  std::shared_ptr<Machine> _machine;
public:
  MachineState(std::shared_ptr<Machine> machine): _machine(machine){}
  virtual std::string status() =0;
  virtual void tick()  {}
  virtual void power() {}
  virtual void open()  {}
  virtual void close() {}
};

class WashingState : public MachineState{
private:
  int _tick = 0;
public:
  WashingState(std::shared_ptr<Machine> machine): MachineState(machine){}
  std::string status() {
    return "(Washing)\n\tdoor:closed;\n\tmachine:on;\n\t" + "tick value: " +
                                                  std::to_string(_tick) + "\n";
  }
  void tick() {
    if( ++_tick == 5400 )
      _machine.setState(std::make_shared<Cooling>(_machine));
  }
  void power(){
    _machine.setState(std::make_shared<Cooling>(_machine));
  }
};

class CoolingState : public MachineState{
private:
  int _tick = 0;
public:
  CoolingState(std::shared_ptr<Machine> machine): MachineState(machine){}
  std::string status() {
    return "(Cooling)\n\tdoor:closed;\n\tmachine:on;\n\t" + " tick value: " +
                                                  std::to_string(_tick); + "\n"
  }
  void tick() {
    if( ++_tick == 120 )
      _machine.setState(std::make_shared<ClosedOffState>(_machine));
  }
};

class OpenOffState : public MachineState{
public:
  OpenOffState(std::shared_ptr<Machine> machine): MachineState(machine){}
  std::string status() {
    return "(OpenOff)\n\tdoor:open;\n\tmachine:off;\n";
  }
  void close() {
    _machine.setState(std::make_shared<ClosedOffState>(_machine));
  }
};

class ClosedOffState : public MachineState{
public:
  ClosedOffState(std::shared_ptr<Machine> machine): MachineState(machine){}
  std::string status() {
    return "(ClosedOff)\n\tdoor:closed;\n\tmachine:off;\n";
  }
  void open() {
    _machine.setState(std::make_shared<OpenOffState>(_machine));
  }
  void power() {
    _machine.setState(std::make_shared<Washing>(_machine));
  }
};
```