```cpp
#include <QSemaphore>

/**
 * PCO 2015 Ex. 14 (c) YTA Yann Thoma
 * Coordination (Brigde Manager) with float numbers
 * Version: 1 (non-FIFO)
 * Objectif: ok (la synchronisation/section critique est garantie)
 * Commentaires:
 *  - ordre FIFO pas respecté.
 *  - famine possible des véhicules lourds
 *    (toujours dépassés par des plus légers)
 */

class Vehicle {
    float weight;
public:
    Vehicle(float weight) : weight(weight) {
        // equiv à this->weight = weight;
    }
    float getWeight() const {
        return weight;
    }
    void stop() {
        // stopped
    }
    void start() {
        // started
    }
};

/*****************************************************************/

class BridgeManagerFloat {
public:
    virtual void access(Vehicle *vehicle) = 0;
    virtual void leave(Vehicle *vehicle) = 0;
};

class BridgeManagerFloat1 : public BridgeManagerFloat {
    QSemaphore *mutex; // protège nbWaiting et currentWeight
    QSemaphore *waitingAccess; // blocage des accès/file d'attente
    int nbWaitingAccess; // nombre de vhc bloqués en attente
    float currentWeight; // poids sur le pont actuellement
    float maxWeight; // poids max du pont

public:
    BridgeManagerFloat1(float maxWeight): maxWeight(maxWeight) {
        // équiv à : this->maxWeight = maxWeight;
        currentWeight = nbWaitingAccess = 0;
        mutex = new QSemaphore(1);
        waitingAccess = new QSemaphore(0);
    }

    ~BridgeManagerFloat1() {
```

```
55          delete mutex;
56          delete waitingAccess;
57      }
58
59      void access(Vehicle *vehicle) {
60          // on garde trace si le vhc a été stopé, pour le restart
61          bool stopped = false;
62          mutex->acquire();
63          // tant qu'il n'est pas possible d'accéder au pont...
64          while (currentWeight + vehicle->getWeight() > maxWeight) {
65              nbWaitingAccess++;
66              mutex->release(); // relâchement: un autre pourra passer ou partir
67              // si pas déjà arrêté, on arrête (éviter les start/stop)
68              if (!stopped) {
69                  vehicle->stop();
70                  stopped = true;
71              }
72              waitingAccess->acquire();
73              // mutex->acquire() pas nécessaire car transmission de mutex!
74          }
75          currentWeight += vehicle->getWeight();
76          // s'il a été arreté, restart
77          if (stopped) {
78              vehicle->start();
79          }
80          mutex->release();
81      }
82
83      void leave(Vehicle *vehicle) {
84          mutex->acquire();
85          currentWeight -= vehicle->getWeight();
86
87          // relâchement éventuel d'un vhc en attente.
88          if (nbWaitingAccess > 0) {
89              nbWaitingAccess--;
90              waitingAccess->release();
91              // transmission de mutex / section critique
92          } else {
93              mutex->release();
94          }
95
96          // sans transmission de mutex: mutex->release(); ici au lieu du esle
97      }
98  };
```