```
#include <QSemaphore>

/**
 * PCO 2015 Ex. 14 (c) YTA Yann Thoma
 * Coordination (Brigde Manager) with float numbers
 * Version: 2 (FIFO - les modifications sont marquées "NEW")
 * Objectif: ok (la synchronisation/section critique est garantie)
 * Commentaires:
 *  - ordre FIFO EST respecté
 *  - pas de famine (passage dans l'ordre)
 *  - implémentation très proche du concept de moniteur
 */

class BridgeManagerFloat2 : public BridgeManagerFloat {
    QSemaphore *mutex; // protège nbWaiting et currentWeight
    QSemaphore *waitingAccess; // blocage des accès/file d'attente
    unsigned int nbWaitingAccess; // nombre de vhc bloqués en attente
    float currentWeight; // poids sur le pont actuellement
    float maxWeight; // poids max du pont

    // NEW FIFO
    QSemaphore *waitingFifo; // blocage en ordre FIFO
    unsigned int nbWaitingFifo; // nombre de vhc en attente de passage
    boolean accessing;

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

        // NEW FIFO
        waitingFifo = new QSemaphore(0);
        nbWaitingFifo = 0;
        accessing = false;
    }

    ~BridgeManagerFloat1() {
        delete mutex;
        delete waitingAccess;
        delete waitingFifo; // NEW
    }

    void access(Vehicle *vehicle) {
        // on garde trace si le vhc a été stopé, pour le restart
        bool stopped = false;
        mutex->acquire();

        // NEW FIFO: le premier passe. Les suivants bloquent.
        if (accessing) { // nota: il y avait un while, mais je pense que if suffit.
            nbWaitingFifo ++;
            mutex->release();
            waitingFifo->acquire(); // mutex transmis
```

```
55          }
56          accessing = true;
57
58          // tant qu'il n'est pas possible d'accéder au pont...
59          while (currentWeight + vehicle->getWeight() > maxWeight) {
60              nbWaitingAccess++;
61              mutex->release(); // relâchement: un autre pourra passer ou partir
62              // si pas déjà arrêté, on arrête (éviter les start/stop)
63              if (!stopped) {
64                  vehicle->stop();
65                  stopped = true;
66              }
67              waitingAccess->acquire();
68              // mutex->acquire() pas nécessaire car transmission de mutex!
69          }
70          currentWeight += vehicle->getWeight();
71          // s'il a été arreté, restart
72          if (stopped) {
73              vehicle->start();
74          }
75
76          // NEW FIFO: on quitte: on libère le suivant dans le fifo s'il existe
77          accessing = false;
78          if (nbWaitingFifo != 0) {
79              nbWaitingFifo--;
80              waitingFifo->release(); // transmission de mutex.
81          } else {
82              mutex->release();
83          }
84      }
85
86      void leave(Vehicle *vehicle) {
87          // NEW FIFO: pas de fifo nécessaire pour le leave
88
89          mutex->acquire();
90          currentWeight -= vehicle->getWeight();
91
92          // relâchement éventuel d'un vhc en attente.
93          if (nbWaitingAccess > 0) {
94              nbWaitingAccess--;
95              waitingAccess->release();
96              // transmission de mutex / section critique
97          } else {
98              mutex->release();
99          }
100
101         // sans transmission de mutex: mutex->release(); ici au lieu du esle
102     }
103 };
```