```
/* *********************************************************
   **********************     Task 8   ********************
   *********************************************************  */

typedef struct {
        float price;
        int item_id;
} item_t;

static item_t *items; // A local storage of items.

static int size;      // Contains MPI::COMM_WORLD.Get_size().
static int rank;      // Contains MPI::COMM_WORLD.Get_rank().
static int num_items; // The size of the items array. At least 1.

#define ALL MPI::COMM_WORLD   // For ease of writing.

// ----------------------------------------------------------------------------

/** Get all items of all processes in any order. Called on exactly one process.
    \param number Returns the number of items in all.
    \return Pointer to all items.
*/
static item_t* get_all_items(int *number){
    /* Your work here. (1) */
}

/** Counterpart to get_all_items. Sends all items to a specific process, called
    on all other processes.
    \param destination The process which shall get all items.
*/
static void send_all_items(int destination){
    /* Your work here. (2) */
}

// ----------------------------------------------------------------------------

/** Compute the average price of the items of all processes.
    Called on each process.
    \return The average price of all items of all processes.
*/
static float compute_average(){
    /* Your work here. (3) */
}
```

```
/* **********************************************************
   **********************   Aufgabe 8   ********************
   **********************************************************  */

typedef struct {
        float price;
        int item_id;
} item_t;

static item_t *items; // Lokale item-Daten des Prozesses.

static int size;      // Gesetzt zu MPI::COMM_WORLD.Get_size().
static int rank;      // Gesetzt zu MPI::COMM_WORLD.Get_rank().
static int num_items; // Die Groesse des items-Feldes. Mindestens 1.

#define ALL MPI::COMM_WORLD   // Zur Schreibvereinfachung.

// ---------------------------------------------------------------------------

/** Liefere alle items aller Prozesse in beliebiger Reihenfolge zurueck.
    Wird in genau einem Prozess aufgerufen.
    \param number Rueckgabe der Gesamtzahl aller items.
    \return Zeiger auf Feld mit allen items.
*/
static item_t* get_all_items(int *number){
    /* Hier vervollstaendigen. (1) */
}

/** Das Gegenstueck zu get_all_items. Sendet alle items an einen bestimmten
    Prozess, wird dazu in allen anderen Prozessen aufgerufen.
    \param destination Der Prozess, der die Daten erhalten soll.
*/
static void send_all_items(int destination){
    /* Hier vervollstaendigen. (2) */
}

// ---------------------------------------------------------------------------

/** Berechne den Gesamtdurchschnittspreis aller items aller Prozesse.
    Wird in jedem Prozess aufgerufen.
    \return Der Gesamtdurchschnittspreis aller items aller Prozesse.
*/
static float compute_average(){
    /* Hier vervollstaendigen. (3) */
}
```