L7: Bitronka

The year is 2025. In Bitronka — the modern empire of retail — there is no longer chaos. There is war. In the heat of the daily struggle for whole-grain rolls, customers create havoc on the store shelves. They swap chips with yogurts, stuff apples between pears. Chewing gum? Best placed next to butter. A can of pineapples? What if it ends up in the dairy section? Worst of all is Mrs. Hermina, who moves AA batteries to the freezer aisle every day because "it's cooler here."

Among the shelves, however, lurks a threat even more dangerous than shelf disarray — pallets. Dozens, hundreds, entire towers of pallets stacked to the ceiling — wobbly and unstable. Maliciously blocking aisles and obstructing emergency exits. For a new employee, one careless step in the aisle can turn fatal — a toppled pallet often means the end of a shift... forever.

In the evenings, when the last customer leaves with a bag full of candy weighed as onions, the night cleaning crew springs into action. They try to restore order. "Those batteries in the freezer again?!" — shouts someone as the echo carries all the way to the fruit section. They know that cleaning the store is a battle against windmills, for chaos will return with the first door opening the next morning. But they don't give up. Each of them randomly selects a pair of products and evaluates whether their location needs changing. If so — the products are moved to the proper (or at least more logical) place. Of course, only if no one trips over a sticking-out pallet along the way. Overseeing the cleaning is the shift manager, who watches in disbelief as his team fights the same, hopeless battle against entropy every night.

Your task is to write a simulation of the night cleaning shift trying to restore order — if only for a moment. **The simulation must be implemented using shared memory and processes.**

Stages:

1. **[6 pts]**
   The program accepts two parameters: `8 <= N <= 256` – the number of products, and `1 <= M <= 64` – the number of workers.
   The store is represented as a one-dimensional array of size `N`, where each index of the array represents a shelf, and the value is the current product (an integer from `1` to `N`).
   - Create the file `SHOP_FILENAME` and map it into the process memory, then initialize the array representing store shelves.
   - Fill it with consecutive integers and shuffle using the provided `shuffle` function.
   - **The use of streams and the `read` function is forbidden.**
   - Before exiting the program, print out the contents of the product array using the provided `print_array` function.
2. **[8 pts]**
   - Create `M` worker processes.
   - Upon creation, each worker prints:
     `[PID] Worker reports for a night shift`
   - After reporting, each worker randomly chooses two distinct indices from the range `[1, N]`, ten times.
   - If the products at those indices are in the wrong order (i.e., value at the first index is greater than at the second), they are swapped to improve shelf order, sleeping for `100 ms` during the swap.
   - After finishing work, the worker process exits.
   - Map an anonymous shared memory region in which mutexes will be stored that protect access to the array values (one mutex per shelf).
   - **Other shared variables should also be stored in this same shared memory region.**
   - Before starting the worker processes, print the contents of the product array.
   - After all worker processes have completed, print the array again along with:
     `Night shift in Bitronka is over`

3. [**6 pts**]
   - After creating the worker processes, create the shift manager process.
   - Upon starting, the manager prints:
     `[PID] Manager reports for a night shift`
   - Every half a second, the manager:
     - prints the current product array,
     - synchronizes the file with its current state,
     - checks if the array is sorted. If it is, he prints:
       `[PID] The shop shelves are sorted`
       and signals workers to end their shift by setting a shared variable in shared memory, then exits.
   - The workers continue until the manager signals the end of their shift.

4. [**5 pts**]
   - In each worker process, add a 1% chance of sudden death right before swapping products (use the `abort` function).
   - Before dying, the worker prints:
     `[PID] Trips over pallet and dies`
   - Living workers and the manager count the number of living (or dead) workers in shared memory.
   - Upon discovering a dead worker, a message is printed:
     `[PID] Found a dead body in aisle [SHELF_INDEX]`
     Note: A body may be discovered twice.
   - After printing the product array state, the manager prints the count of alive workers:
     `[PID] Workers alive: [ALIVE_COUNT]`
   - When no workers remain alive, the manager prints:
     `[PID] All workers died, I hate my job`
     and exits.