eXascale Infolab

Handout

Mini-project MP01: The Fribourgeois bakery

Context

The Fribourgeois bakery is located in downtown Fribourg. The bakery has an inventory composed of all ingredients a baker needs (flour, oil, baking powder). The bakery also offers a variety of bread; each has a dedicated shelf in the bakery's display racks. Each summer, the bakery welcomes apprentices from all over Switzerland, and it gets a bit messy. The baker of this Fribourgeois bakery came to you to help her find some strategies to organize the bakery. Since she is familiar with computer science, she asked you to implement the solution in C or C++.

Learning to make bread

The baker wants all her apprentices to learn bread recipes. However, she has limited inventory resources (flour, oil, baking powder) that are added routinely, making the apprentices race to get to these resources. Therefore, she adopted the following strategies in the previous days:

- Strategy "arrival order": When apprentices arrive, they have to check their arrival time. The baker would then sort their arrival time and use that order to access the inventory.
- Strategy "fast learners": After a few days of apprenticeship, the baker noted that some of her apprentices learned faster than others. Therefore, the baker tried to sort them according to their learning speed and use that order to access the inventory.
- Strategy "step-by-step": One other strategy, the baker tried to allow only a small amount of time for each apprentice to access the inventory.

The baker tried these strategies, but she was not fully satisfied. Therefore, she would like you to implement one of them and show her when you expect it to perform well and when you expect it to perform poorly.

Too good to go

Part of managing a bakery is managing the leftover. The baker wants to teach this skill to her apprentices. One or two hours before closing the bakery, she has to choose which types of bread to donate. The choice has to be wisely made because she does not want to donate bread that is likely to be requested by customers in the last opening hours. She usually opts for one of the solutions below and she wants you to pick one.

UNI FR

- Strategy “donate the old bread”: The baker donates all types of bread that have been in the bakery the longest.
- Strategy “Second chance”: The baker gives a second chance to types of bread that have been in the store the longest but have been sold recently.
- Strategy “Not recently sold”: The baker donates the types of bread that have not been sold recently.

Additional Features

Implement one of the following additional features:

- The baker and apprentices would like to make a cuchaule together. The baker puts the needed ingredients on the table, and one of the apprentices uses them for a step of the recipe[1]. The baker and the apprentices are represented as threads. Implement a solution to synchronize them.
- The bakery has some loyal customers who regularly buy their bread from the bakery. The baker has created n chairs for them. If there are no customers present, the baker falls asleep. When a customer arrives, he has to wake up the sleeping baker. If additional customers arrive while the baker is serving a customer, they either sit down (if there are empty chairs) or leave the shop (if all chairs are full). Implement a solution to synchronize the customers with the baker.

Deliverables:

Deliverables to be ready for Fri**, 27. May 2022** to be uploaded on **Moodle**:

- A repository on Gitlab with a runnable C or C++ code with a makefile, testing scripts, and a readme file.
- A short three pages documentation per group.
- Preparation of a final presentation per group (3 min/Person).

1. Minimal Feature Set

a) Bakery initialization: Choose data structures to represent the bakery’s inventory and the display racks. Each item in the store is associated with the available amount. Note that the data structures can be dynamic.

Example: There are three kilograms of flour in the bakery.

b) Implement your solution for “Learning to make bread” such that each apprentice learns the recipe of Cuchaule and access the inventory as the baker wants. Hint: scheduling, reader-writer problem. A stopping criterion should be used such as clock time, number of breads made by an apprentice, etc.

c) Implement your solution for “too good to go” such that you manage the leftover of the bakery whenever needed. Hint: Page Replacement

d) Implement an additional feature of your choice.

e) Provide automated testing in the form of another program or script showing all the functionalities mentioned above.

[1] https://www.terroir-fribourg.ch/fr/recettes/recette-cuchaule

2. Implementation decisions

- *Index:* Efficient search for the items in the inventory (e.g., binary search).
- *Data structure Size:* How many entries can the inventory have? Can the size grow?
- *Simultaneous access:* How many apprentices can access the inventory simultaneously?
- *Synchronization:* How to synchronize between apprentices?

3. Scenarios

- Apprentices add the same item to the inventory.
- Apprentices access the same item in the inventory.
- An apprentice is adding an item, and another is retrieving it.

4. Project schedule

Due by 12th April: Implement your solution of the minimal features a) inventory initialization and b) learning to make bread.

Due by 26th April: Implement your solution of the minimal features c) "too good to go" and manage the corner cases. Prepare a script to test each scenario in section 3.

Due by 10th May: Implement your additional feature and start cleaning your code and scripts.

Due by **27th May**: Submit your report and presentation.

The schedule can be used as a timeline for the project. Only the date in bold (27th May) is a hard deadline.

5. Evaluation Key (subject to change)

The following aspects of your work will be taken into account for the final mark with the according weights.

Implemented Features	**50%**
Minimal Feature Set	30%
Testing program / script	10%
Additional Features	10%
Code Quality	**30%**
Simple and Concise Code	10%
Commented Code	10%
Architecture and Helper files (Makefile)	10%
Documentation	**20%**
Final 3 min/Person Presentation	10%
Time	5%
Content	5%
Short Documentation (3 Pages)	10%
Usage of Store and Testing	5%
Architecture and Decisions	5%