Übungsblatt 9

Abgabe bis spätestens 25.01.2024 in Stud.IP

Allgemeiner Hinweis

Für dieses Übungsblatt benötigt ihr das in StudIP zur Verfügung gestellte Archiv `ueb9-vorgabe.zip` mit Vorgabedateien.

Aufgabe 1 (3 Punkte)

Vier Freunde beschließen, einen Mittagstisch für Senioren zu eröffnen. Sie bieten drei Gerichte an: „Gemüseauflauf“, „Wraps“ und „Pizzabrötchen“. Als Grundzutaten kaufen sie täglich vom Großmarkt Käse, Gemüse und Teig.

Der Auflauf besteht aus Gemüse und Käse und wird von Jonas zubereitet. Hava macht die Wraps aus Gemüse und Teig und Lara verwendet Teig und Käse zum Zubereiten der Pizzabrötchen.

Yin ist die Fahrererin: sie kauft die Zutaten ein und liefert die fertigen Gerichte aus. Auf jeder Fahrt kann sie nur eine Kiste mit Zutaten transportieren, da sie mit einem E-Roller unterwegs ist. Das Startkapital der Freunde reicht gerade für zwei Kisten. Die Vorauskalkulation für das Unternehmen der vier Freunde sieht vor, dass der Erlös aus dem Verkauf eines Gerichts um fünf Prozent über den Materialkosten liegt. Sobald das angesparte Kapital für den Erwerb weiterer Kisten ausreicht, wird Yin damit weitere Kisten erwerben (es können aber immer nur vollständige Kisten für jeweils 10 Euro erworben werden. Da Yin nicht über den Lagerbestand Buch führen muss, sorgt sie immer dafür, dass sich eine gekaufte Grundzutat von der jeweils vorher gekauften Grundzutat unterscheidet.

Die Organisation des Unternehmens ist allerdings nicht perfekt: Yin weiß gar nicht, welche Zutaten Jonas, Hava oder Lara jeweils gebrauchen können. Wenn Yin eine Kiste besorgt hat, stellt sie sie im Lagerraum ab, holt eine zweite Kiste mit anderen Zutaten als in der ersten Kiste waren, stellt auch sie im Lager ab usw. Immer, wenn sie eine neue Kiste im Lager abgestellt hat, schaut sie, ob bereits neue Speisen fertig sind. Wenn ja, nimmt sie auf jeder Fahrt zum Markt eine Speise mit, um sie dort zu verkaufen. Das bringt wiederum neue Einnahmen, mit denen er weitere Kisten kaufen kann.

a) Implementiert eine funktionierende Lösung mit Semaphoren unter Verwendung der Klasse `Sema` aus `sema.hh`. Ihr findet in `aufgabe1/shop.cc` ein Gerüst für euren Code und ein entsprechendes `Makefile`. Jonas, Hava, Lara und Yin werden jeweils durch einen eigenen Thread modelliert. Der Thread Yin nutzt zum „Einkaufen“ den ZutatenGenerator aus `zutatengenerator.hh`.

   Die Funktion `herstellung()` dient zur Implementierung der Threads für Jonas, Hava und Lara und wird mit entsprechend unterschiedlichen Kombinationen aus Grundzutaten parametrisiert. Achtet bei eurer Modellierung darauf, dass Jonas, Hava und Lara die Kisten mit den Zutaten im Lager einzeln prüfen müssen. Yin gibt keine weitere

Information darüber, welche Zutaten im Lager vorhanden sind. (Das heißt, es darf z. B. keine Variable `gemuese_teig` geben, die von Yin gesetzt wird.)

Arbeitet für die Synchronisierung nur mit den Funktionen `P()` und `V()` aus der Klasse `Sema`. Andere Mechanismen für die Synchronisierung von Threads sind nicht erlaubt. Hinweis: Die vorliegende Fassung wird ohne Schutz der kritischen Abschnitte bei nebenläufigen Zugriffen nicht funktionieren, ihr müsst euch also von Beginn an Gedanken über den korrekten Einsatz der Semaphoren machen.

b) Bewertet kurz aber präzise eure Umsetzung. Hat sie Schwächen (z. B. aktives Warten)?

Aufgabe 2 (4 Punkte)

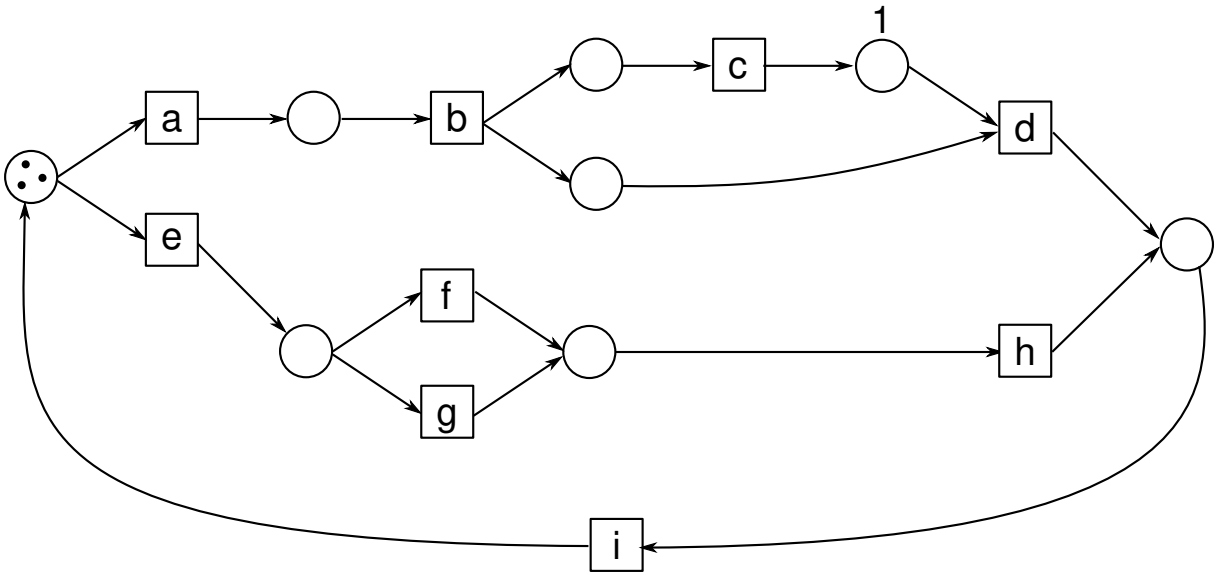
Modelliert das Problem aus Aufgabe 1 als Petrinetz entsprechend der in der Vorlesung vorgestellten Definition. Dokumentiert eure Lösung.

Achtet auf eine saubere Modellierung, zum Beispiel sinnvolle Tokens, die zum Beispiel das Geld modellieren. Diese dürfen nicht zwischendurch kurz „aufgespalten“ und dann später wieder zusammengefügt werden. Es darf je Grundzutat nur eine Transition zu deren Kauf geben. Der Kauf muss sequentiell erfolgen, d. h. es darf nicht möglich sein, dass zwei Kisten mit Zutaten zeitgleich gekauft werden. Es sollten möglichst keine überflüssigen Hilfskonstruktionen (Stellen und Transitionen) eingeführt werden, die in der „Geschichte“ gar nicht vorkommen. Nutzt die Möglichkeiten, die euch Petrinetze bieten. Hinweis: Wenn ihr einen Petrinetz-Simulator[1] verwendet, benennt den verwendeten Simulator und legt eurem Lösungsvorschlag bitte das Petrinetz als Abbildung und die Quelldatei für die Simulation bei (bitte nicht den Simulator selbst). Fehlende Modelliermöglichkeiten der Tools solltet ihr ggf mit einem Grafikprogramm ergänzen.

Aufgabe 3 (3 Punkte)

Wie könnten die Synchronisationsbedingungen des folgenden Petrinetzes mit Semaphoren ausgedrückt werden? Modelliert dafür in Pseudo-Code einen Thread für jede Transition (a bis i), jeder Übergang soll mit einer oder mehreren Semaphoren versehen werden, die die Ausführungslogik des Petrinetzes modellieren. Denkt daran, eine korrekte Initialisierung für die verwendeten Semaphoren anzugeben.

[1] Zum Beispiel PIPE2 `https://sourceforge.net/projects/pipe2/`. Einen Browser-basierten Online-Editor findet ihr zum Beispiel unter https://apo.adrian-jagusch.de/.



Abgabe

Bis 23:59 Uhr am 25.01.2024 digital in StudIP. **Es gelten die vereinbarten Scheinbedingungen (siehe StudIP).** Bitte beachtet unsere ergänzenden Hinweise ebenda.

Ladet die abgaberelevanten Dateien in DoIT hoch.

Eure Ansätze und der gewählte Lösungsweg müssen nachvollziehbar sein. Achtet insofern auf eine saubere Dokumentation im Quelltext und im abgegebenen PDF-Dokument. Benennt alle von Euch verwendeten Quellen, auch Zusammenarbeit mit anderen Gruppen und verwendete Unterlagen aus früheren Jahrgängen.

Programmieraufgaben sind im Zweifel in C++20 zu entwickeln. Die Korrektheit der Lösung bzw. deren Grenzen sind grundsätzlich in der Abgabe nachzuweisen. Dies geschieht neben der Dokumentation des Programmcodes in den Quelldateien und zusätzlich in dem abgegebenen PDF-Dokument mit der Lösungsbeschreibung durch geeignete Tests, deren Auswahl und Eignung begründet werden müssen. Als **Referenzplattform** gelten die Linux-Rechner im Rechnerpool in MZH E0.