Segunda avaliação – PLC 22.2
Entrega do código no Classroom até as 13h do dia 19/04/2023
Apresentação do código pela dupla em sala no dia 19/04/2023, no horário da aula

1. [Variáveis de condição] Você foi contratado para implementar um sistema de administração aeroportuário que facilite o trabalho dos controladores de voo.Para tal fim, o seu sistema deve ser capaz de automaticamente avisar aos aviões quando eles podem decolar ou aterrisar.

O programa irá receber como entrada uma quantidade N (especificada pelo usuário) de aviões esperando para sair. Depois disso, ele irá receber, para cada avião, a hora de saída do mesmo (para fins de teste, essa hora de saída será especificada em milissegundos após o início do programa). Após ter recebido os aviões que irão decolar, o programa irá receber um número M (também especificado pelo usuário) de aviões que irão chegar, cada um com a sua hora esperada de chegada. Finalmente, o usuário deverá inserir um número K, correspondente ao número de pistas disponíveis no aeroporto.

Quando chegar a hora de um avião decolar ou aterrisar, o programa deve verificar se há uma pista livre. Se houver, o avião irá utilizar a pista. O processo de decolagem ou de aterrissagem de um avião ocupa a pista por 500 milissegundos. Caso não haja uma pista livre, o avião deverá aguardar até que haja.

Quando um avião consegue decolar ou aterrissar com sucesso, o programa deve imprimir o horário esperado de saída do avião, o horário real, e o atraso que ocorreu. Além disso, para evitar prejuízos para os aeroportos, seu programa deve minimizar os atrasos - portanto, quando uma pista for ocupada,
deve ser dada prioridade ao avião cujo horário de saída é mais cedo.

Por motivos de eficiência, o programa deve ser implementado sem uso de espera ocupada, e quando um avião ser notificado de que há uma pista livre, deve ser realizada uma única chamada - ou seja, não é permitido buscar no conjunto de aviões para encontrar aquele que deve sair mais cedo.

2. [Executors e thread pools] Em uma colméia, operários fazem tarefas. Tarefas possuem 3 propriedades: id da tarefa, tempo necessário para a resolução da tarefa e as tarefas que precisam estar prontas para que ela seja iniciada. A rainha tem uma fila de tarefas (a ordem da fila é a ordem na entrada) e sempre que uma tarefa pode ser realizada ela passa a mesma para que algum operário a faça, assim que estiver livre. Caso alguma tarefa não possa ser iniciada ela irá para o fim da fila. O programa acaba quando todas as tarefas estiverem concluídas.

Entrada:
O programa irá pedir um número "*O*" de operários, depois um número "*N*" de tarefas a serem realizadas.
As próximas *N* linhas serão as propriedades de cada tarefa no formato a seguir: o primeiro número será *id* da tarefa (é garantido que os *id* vão de 1 até *N*); o segundo será o tempo para a resolução (em ms) e os restantes (caso existam) os *id*s de tarefas de que esta depende.

Saída: a cada tarefa realizada deve se imprimir "tarefa id feita", sendo id o id da tarefa.

Exemplo

Entrada:
1 4
1 200 2 4
4 30
3 50 2
2 100

Saída:
tarefa 4 feita
tarefa 2 feita
tarefa 1 feita
tarefa 3 feita

3. A máquina de refrigerante do Rei do Hambuguer, funciona de uma forma bastante simples. A todo momento, existem clientes enchendo seus copos de refrigerantes, sendo estes de 3 tipos: Pepise-Cola, Guaraná Polo Norte e Guaraná Quate. Cada cliente leva 1000 ms para encher seu copo com o refrigerante. Porém a maquina só pode ser utilizada por uma pessoa por vez, então se um cliente está enchendo o copo com Guaraná Quate, o outro cliente que deseja Guaraná Polo Norte tem que aguardar a máquina estar disponível para ele. Cada cliente enche seu copo com 300 ml de refrigerante e a máquina suporta apenas 2000 ml de cada refrigerante, ou seja, 6000 ml no total. Para evitar transtorno, sempre que um refrigerante possuir menos que 1000 ml, ele é reposto com um refil de 1000 ml. Para repor o refrigerante, a máquina não pode estar sendo utilizada por nenhum cliente e demora 1500 ms para reposição. Dado o número de clientes que irão estar o tempo todo querendo o refrigerante Pepise-Cola, Guarana Polo Norte e Guaraná Quate, implemente a máquina de refrigerante do Rei do hambuguer utilizando o conceito de Variáveis Mutáveis em Haskell. Sempre que alguém estiver utilizando a máquina de refrigerante, apresente as seguinte informações.

Caso seja para abastecer a máquina: "O refrigerante X foi reabastecido com 1000 ml, e agora possui Y ml", onde X é o nome do refrigerante e Y a quantidade que possui do mesmo após ele ser reabastecido.

Caso seja um cliente, informe: "O cliente N do refrigerante X está enchenco seu copo", onde N é o número do cliente para seu respectivo refrigerante e X o nome do refrigerante.

4. Utilizando coroutine na linguagem Lua, simule uma batalha pokemon entre Pikachu e Raichu, onde Pikachu possui 800 de HP e Raichu possui 1000 de HP. Os dois possuem a mesma lista de ataques com a mesma probabilidade de usar cada um deles. Você deve simular a batalha, onde cada um ataca em um turno (é sua escolha quem começa atacando) e você deve informar para cada turno qual pokemon utilizou qual ataque e quanto de HP os dois ainda possuem. A batalha acaba quando algum pokemon possuir HP menor ou igual a 0. Para utilizar cada ataque, você deve gerar um número randomicamente entre 1 e 20. Caso o número gerado seja entre 1 e 10, o ataque que será utilizado é o Choque do trovão, causando 50 de dano. Caso seja entre 11 e 15, o ataque que será utilizado é o Calda de ferro, causando 100 de dano. Caso seja entre 16 e 18, o ataque utilizado será o Investida Trovão, causando 150 de dano. E por último, caso seja entre 19 e 20, o ataque utilizado será o Trovão, causando 200 de dano.