Segunda Avaliação – PLC-CC (IF686) – 23.1
Prof. Márcio Lopes Cornélio
Entrega do código no Classroom até as 8h do dia 26/09/2023
Apresentação do código pela dupla no laboratório no dia 26/09/2023, no horário da aula, com assinatura da ata.

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. [Mvar e Tvar] Você ficou responsável por implementar um sistema de monitoramento de um estacionamento. Dado um estacionamento onde irão chegar N carros em uma determinado dia (não importa a ordem de chegada dos mesmos), onde esse estacionamento possui K vagas, você terá que informar qual carro pegou qual vaga em qual instante de tempo, e ao carro sair, deve-se informar também qual vaga foi liberada por qual carro em qual instante de tempo. N e K são entradas do usuário. O instante de tempo inicial é 0, e inicialmente o estacionamento está com todas as K vagas disponíveis. Como todo estacionamento, 10% das K vagas são destinadas a pessoas com deficiência (PCD), os únicos carros que podem ocupar estas vagas são os 20% primeiros (de 1 até 20% de N). Todo carro identificado de PCD pode parar em qualquer vaga que estiver disponível no estacionamento no momento, porém os carros não identificados de PCD podem parar apenas nas vagas não preferenciais que estão disponíveis. Dado que cada carro quando para em uma vaga leva 1000ms para deixar a mesma, menos os carros identificados para PCD quando param em vagas não especiais, quando isto ocorre 1500ms é o tempo em que o carro fica ocupando a vaga. Implemente o monitoramento desse sistema em Haskell utilizando tanto o conceito de MVar como o de TVar (observe que são necessárias 2 implementações distintas) e diga qual foi a dificuldade de cada uma e porque.

4. [Java e Haskell] Alguns dos problemas clássicos de concorrência são: pingping e o pingpong.

pingping - (comunicação assíncrona) sendo o ‘n’ o número de mensagens, uma thread manda ‘n’ mensagens indefinidamente e a outra consome as mensagens assim que recebidas. O programa acaba quando que todas mensagens são recebidas;

pingpong - (comunicação síncrona) sendo o ‘n’ o número de mensagens, uma thread manda uma mensagem espera a outra receber para poder mandar mais uma. O programa acaba assim que todas mensagens são recebidas;

Implemente os dois problemas em Java e em Haskell, e comente sobre a escolha da forma usada para garantir a coerência dos dados e a corretude da questão (se for MVar ,TVar , Lock ,Atomic, synchronized …) e quais as dificuldades ou facilidades encontradas para modelar cada um dos problemas tanto em Haskell quanto em Java.