4ª Lista de Exercícios de Paradigmas de Linguagens Computacionais
Professores: Fernando Castor, Márcio Cornélio e Paulo Borba
Monitores:
Ciência da computação: Alberto Rodrigues Costa Junior (arcj), Diego Correia Aragao (dca), Luana Martins dos Santos (lms7), Luiz Fernando Sotero (lfss2)
Hugo Bessa (hrba), Joao Victor de Figueiredo Leite (jvfl), Vítor Antero (vham).

CIn-UFPE – 2013.1
Disponível desde: 27/08/2013
Entrega: 10/09/2013

A lista deverá ser respondida **em dupla**. A falha em entregar a lista até a data estipulada implicará na perda de 0,25 ponto na **média** da disciplina para os membros da dupla. Considera-se que uma lista na qual **menos que 5** das respostas estão corretas não foi entregue. A entrega da lista com **pelo menos 9** das questões corretamente respondidas implica em um acréscimo de 0,125 ponto na média da disciplina para os membros da dupla. A resolução da **última questão (11)** e da **primeira (0) é obrigatória**. Se **qualquer situação de cópia de respostas** for identificada, os membros **de todas as duplas envolvidas** perderão 0,5 ponto na média da disciplina. O mesmo vale para respostas obtidas a partir da Internet. As respostas deverão ser entregues **exclusivamente em formato texto ASCII** (nada de .pdf, .doc, .docx ou .odt) e deverão ser enviadas para o monitor responsável por sua dupla, **sem** cópia para o professor. Devem ser organizadas em arquivos separados, um por questão, entregues em um único formato compactado, ou seja, um único arquivo zipado contendo as respostas para todas as questões.

0) De acordo com a Wikipedia (http://pt.wikipedia.org/wiki/Problema_das_oito_damas) :

> *"O problema das oito damas é o problema matemático de dispor oito damas em um tabuleiro de xadrez de dimensão 8x8, de forma que nenhuma delas seja atacada por outra. Para tanto, é necessário que duas damas quaisquer não estejam numa mesma linha, coluna, ou diagonal. Este é um caso específico do Problema das N damas, no qual temos N damas e um tabuleiro com N×N casas(para qualquer N ≥ 4)."*

Construa um programa que, dado um número *N*, resolve de forma paralela o problema das N damas. Desenvolva também uma versão sequencial e compare os desempenhos quando variamos o valor de *N* e o número de threads empregadas. Seu programa deve produzir como resultado as coordenadas onde as damas deverão ser colocadas. Por exemplo, para um tabuleiro 4×4, o programa pode produzir como saída:

(1,2)
(2,4)

(3,1)
(4,3)

Onde o primeiro item de cada par representa a linha e o segundo a coluna.

1)Usando **MVars**, implemente o algoritmo Quicksort em haskell de forma que ele possa se beneficiar do uso de threads. Após implementado, varie o tamanho da entrada para o algoritmo e o número de threads utilizadas na sua execução e, explicitando os valores usados na resposta, explique o porquê das mudanças de desempenho do código dada a variação no número de threads e no tamanho da entrada.

2) Implemente uma solução em Haskell, utilizando **MVars**, para uma variação do problema do Produtor e do Consumidor onde há vários (dois ou mais, conforme definido pelo **usuário**) produtores e cada um tem seu próprio buffer (um por produtor). Os buffers de todos os produtores têm a mesma capacidade (pelo menos 10 posições). Além disso, o sistema tem vários consumidores (mesma quantidade que a de produtores). Cada produtor coloca itens produzidos apenas em seu buffer particular mas consumidores podem consumir itens de qualquer buffer. Fora isso, as mesmas regras do Produtor-Consumidor básico valem: consumidores não podem consumir de um buffer vazio, embora possam tentar consumir de outro buffer, e produtores não podem produzir além da capacidade de seus buffers. Além disso, o sistema nunca pode entrar em deadlock.

3) Uma lanchonete consiste em três caixas e uma fila única de atendimento. Se não há nenhum cliente na fila os caixas esperam até que clientes entrem na lanchonete. Se um cliente entra e vê que a lanchonete esta muito cheia (definido por uma variável inicial que limita o tamanho da fila), ele vai embora. Senão ele entra na fila e espera ser atendido. Seu programa deve rodar até que um número máximo de clientes a ser atendidos (definido inicialmente) tenha sido atingido.

- Cada caixa deve ser implementado como uma Thread distinta, quando um caixa esta livre e há clientes na fila, ele irá atender o primeiro da fila e o cliente sairá da lanchonete(liberando um espaço na fila).
- Cada cliente deve ser implementado como uma Thread distinta.
- A lista de clientes esperando deve ser mantida em uma MVar.
- Clientes chegam a cada 50ms a 500ms definido aleatoriamente.

Escreva informalmente as propriedades de segurança e vivacidade para este problema e mostre que seu programa satisfaz essas propriedades.

4)Dança das cadeiras é uma brincadeira infantil muito comum. Nela um número X de crianças dançam ao redor de X-1 cadeiras até que a música pare. Nesse momento todas as crianças tentarão sentar numa das cadeiras e a que não conseguir sai da brincadeira.
Desenvolva um programa que simule uma partida de dança das cadeiras. Seu programa deve receber uma lista com os nomes dos participantes. A cada rodada deve imprimir quem saiu da brincadeira e ao final da execução deve imprimir quem foi o vencedor. Seu programa deve usar TVars para garantir o acesso consistente das

crianças às cadeiras.

5) Você está programando um servidor que cria partidas de dominó online. Em cada partida, apenas 4 pessoas jogam. Implemente o servidor de tal forma que as partidas são criadas a cada 4 requests de jogadores. Um novo jogador tenta jogar a cada 60ms ou 120 ms e o servidor consegue criar apenas 2 partidas por segundo. Se um jogador tentar entrar e a fila de espera do servidor estiver cheia (definida pelo usuário), ele desiste e vai jogar paciência. Imprima no console toda vez que uma pessoa tentar jogar e todo jogo que o servidor formar. Além disso, relate no final da execução do programa a quantidade de jogos formados e a quantidade de usuários que não conseguiram jogar! Seu programa deve rodar por um tempo fixo, definido pelo usuário. Utilize **TVars** na solução do problema.

Exemplo de saída do programa (fim):

User: Happy guy gonna play dominoes!
(...)
User: Happy guy gonna play dominoes!
(...)
User: Sad dude gonna play solitaire!
(...)
User: Sad dude gonna play solitaire!
(...)
Server: match created
(...)
User: Happy guy gonna play dominoes!
(...)
User: Happy guy gonna play dominoes!
(...)
20 matches were created!
But 28 went to play solitaire =(

Acima, o símbolo “(...)” indica que pode haver outras linhas entre a anterior e a posterior.

6) Em uma biblioteca, a tarefa de Hazel é retirar os livros que os leitores deixavam sobre as mesas. É uma regra da biblioteca, para evitar que os livros estivessem em prateleiras erradas. Em cada mesa, os leitores podem deixar os livros que leram (no mínimo 1 livro e no máximo 10 livros). O leitor entra na biblioteca, escolhe os livros e uma mesa e lê cada livro em 2 segundos. Em seguida retira outros livros de prateleiras da biblioteca deixando os já lidos na mesa e escolhe uma mesa novamente. É possível que a mesa anterior seja escolhida de novo. Se a mesa escolhida não estiver disponível, o leitor busca novos livros para ler. Uma mesa está disponível quando está sem livros. Considerando a situação descrita, implemente um programa para simular essa biblioteca usando **MVars**.

A entrada do programa de simulação será o número de mesas da biblioteca e a

quantidade de leitores. A saída do programa será o total de livros retornados às prateleiras por Hazel. O tempo de simulação deverá ser definido previamente.

7) Resolva a questão 7 da Lista 3 (a questão das irmãs) em Haskell, utilizando MVars. Utilize os métodos não bloqueantes para trabalhar com MVars.

8) Seu Zé possui uma fábrica de chocolates muito boa, só que infelizmente, a demanda da fábrica é muito grande e ele não consegue supri-la. Sabendo disso, Seu Zé, muito ganancioso, decide automatizar o processo com máquinas e estuda a compra de dois tipos de máquinas. O primeiro tipo máquina é uma chocolateira. Ela recebe os ingredientes brutos e fabrica o chocolate, entregando 100 chocolates a cada vez que consegue acesso à esteira. O segundo tipo trata-se de uma embaladeira, que recebe chocolates prontos e coloca-os na embalagem, deixando-os prontos para a venda. A embaladeira consegue embalar 80 chocolates a cada vez que consegue acesso a esteira. A esteira da fábrica possui espaço para apenas 400 chocolates.

Como as máquinas são muito caras, Seu Zé convenceu o professor de PLC do CIn a passar um simulador da fábrica como uma questão da lista 4 da disciplina.

Esse simulador deve receber como entrada a quantidade de máquinas do tipo 1 e do tipo 2 e deverá rodar até que sejam fabricados 100000 chocolates. Deve gerar um log ao final da execução da quantidade de vezes que a esteira ficou cheia ou vazia, para que Seu Zé tenha uma ideia de quão improdutiva está sendo a configuração testada. Seu programa deve usar TVars para garantir a concorrência das máquinas.

9) Você é um Nephalem e está preparado para travar a batalha final contra Malthael, o Anjo da Morte. Lutando junto com você estão Tyrael, ex-Arcanjo da Justiça e seu follower. Felizmente, Malthael está enfraquecido e não consegue lutar igualmente com vocês, restando a ele retardar sua morte ao diminuir o dano dos heróis. Sua magia Enfeeble, diminui o dano de uma unidade inimiga para 30% durante 3 ataques. Malthael sempre escolhe um inimigo aleatório e, se o inimigo já estiver sob efeito desta magia, aumenta a quantidade de ataques que ele executará com dano diminuído.

Cada personagem é uma thread distinta. Durante a execução do programa, imprima todos os eventos, ou seja, feitiços lançados por Malthael e golpes desferidos pelos heróis. No fim do programa, informe quanto cada herói feriu o Anjo da Morte, até a sua execução.
O HP de Malthael nunca pode ser menor do que zero. Malthael começa com 500000 hitpoints. Cada golpe do Nephalem causa 150 de dano. Golpes de Tyrael causam 105 pontos de dano e golpes do follower causam 75. Utilize **TVars** na solução do problema.

Exemplo de saída do programa (fim):

Follower: hit Malthael for 75 hitpoints
Tyrael: hit Malthael for 105 hitpoints
Nephalem: hit Malthael for 150 hitpoints
Malthael: Follower enfeebled for three more rounds
Follower: hit Malthael for 75 hitpoints

Tyrael: hit Malthael for 105 hitpoints
Nephalem: hit Malthael for 105 hitpoints
Malthael: I yieeeeeld!
Follower: inflicted a total of 119175 hitpoints to Malthael
Tyrael: inflicted a total of 160720 hitpoints to Malthael
Nephalem: inflicted a total of 220105 hitpoints to Malthael

10) Jason faz aluguel de carros. Existem 3 preços para o carro escolhido considerando os acessórios que eles podem ter.
- O carro com acessório 1 custa £40 a hora.
- O carro com acessório 2 custa £60 a hora.
- O carro com acessório 3 custa £100 a hora.

Os clientes de Jason podem gastar valores de 500 a 1000 nos aluguéis cada vez que aluga um carro.
Eles ficam com o carro no mínimo 1 hora e no máximo o quanto podem gastar em um aluguel.
(Exemplo: Se a cliente Annabeth puder gastar 500 libras e escolher o carro com acessório 1, Annabeth ficará com o carro por 12 horas). Se o carro não estiver disponível, o cliente tenta alugar o mesmo carro até conseguir. Após isso ele espera um tempo definido previamente para alugar outro carro.
Simule um dia de trabalho de Jason com um programa Haskell utilizando **TVars**.
Este programa requisitará ao usuário uma quantidade de clientes a serem atendidos e a quantidade de carros existentes em cada faixa de preços na empresa de Jason. O valor que cada cliente está disposto a gastar é determinado aleatoriamente (entre 500 e 1000£). Ao término do programa, Imprima quanto Jason ganhou na simulação.

11) Dois encanadores, Márcio, um homem baixinho com barrigão, bigode preto, sempre com seu boné vermelho preferido, e seu irmão Luiz, um rapaz alto, magro, com bigode tão grande quanto o do seu irmão e com um boné verde, estão trabalhando **[1]** na manutenção da tubulação de um reino.
No ambiente de trabalho, existem 10 canos que precisam ser mantidos em ordem. Cada cano pode sofrer um entre três problemas. Para solucionar o primeiro problema é necessário uma fermenta A, para solucionar o segundo é necessário uma ferramenta B, e para solucionar o terceiro são necessárias as duas ferramentas. **[2]** Como encanadores são mal pagos nesse reino, os irmãos tem apenas uma de cada ferramenta.
Faça uma simulação na qual cada cano tenha 20% de chance de ter algum problema aleatório (não cumulativo com 40% de chance de ser o problema 1, 40% de ser o problema 2 e 20% de ser o problema 3) a cada X milissegundos, sendo X um valor entre 200 e 400 milissegundos. Considere que que cada encanador leva 80 milissegundos para resolver qualquer um dos problemas. Caso um encanador tente consertar um problema cuja ferramenta esteja indisponível, ele deve ir para outro cano.
Imprima na tela cada vez que um encanador tentar consertar um cano e no final do expediente (10 segundos) imprima quantos canos foram concertados. Utilize uma thread para cada encanador, e uma thread para cada cano.

Use TVars para resolver esta questão.

Observações:
[1]: Para trabalhar, cada um usa sempre um macacão azul, com uma blusa da cor do seu chapéu.
[2]: Os problemas são os seguintes:

1. Uma tartaruga acaba entrando dentro do cano, necessitando de um desentupidor para removê-la.
2. Uma planta carnívora entra dentro da tubulação necessitando de um lança chamas para mata-la.
3. É um problema demasiadamente complexo para ser descrito no escopo da questão, mas é de senso comum que ele pode ser solucionado com as ferramentas ue solucionam os defeitos um e dois.