UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE CIÊNCIAS EXATAS E DA TERRA
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA
DIM0124 – PROGRAMAÇÃO CONCORRENTE

Trabalho Prático

Programação concorrente em Go

Objetivo

O objetivo deste trabalho é colocar em prática o projeto e a implementação de programas concorrentes utilizando a linguagem de programação Go (https://go.dev).

1 Problemas

Problema 1: Um Jogo de Tênis

O *tênis* é um esporte de origem inglesa, disputado em quadras geralmente abertas e do qual participam do jogo dois oponentes ou duas duplas de oponentes. A quadra é dividida em duas meia-quadras por uma rede e o objetivo do jogo é rebater uma pequena bola para além da rede (para a meia-quadra adversária) com ajuda de uma raquete. Para marcar um ponto, é preciso que a bola toque no solo em qualquer parte dentro da quadra adversária, fazendo com que o adversário não consiga devolver a bola antes do segundo toque ou que a devolva para fora dos limites da outra meia-quadra.

A estrutura de pontuação de um jogo de tênis consiste basicamente na tríade *game–set–match*. Um *game* é uma sequência de pontos marcados por um jogador, ganhando aquele que primeiro marcar pelo menos quatro pontos no total e pelo menos dois pontos a mais que o adversário. Um *set* é um conjunto de *games*, em geral vencendo o jogador que ganhar pelo menos seis *games* e pelo menos dois *games* a mais que o oponente. Por fim, uma partida como um todo é denominada *match*, consistindo de uma sequência de *sets* e sendo vencedor o jogador que ganhar três de cinco *sets*.

Escreva um programa em Go que simule um jogo de tênis com dois jogadores, cada um deles podendo ter dois possíveis estados, a saber (i) esperando para receber a bola ou (ii) mandando a bola de volta para o adversário. Para simular a dinâmica do jogo, você pode utilizar facilidades providas pelo pacote `math/rand`[1] para determinar, de forma aleatória, se o jogador em questão perdeu a bola ou não – neste último caso marcando ponto o jogador adversário. Por questões de simplicidade, admita que o *match* possui um único *set* composto de um único *game*, ganhando o jogo o jogador que alcançar um número de pontos fixo P desse *game*.

[1] Para este caso, é necessário garantir que os valores gerados são diferentes a cada execução do programa.

Extra: Torne o programa mais interessante (e realista) permitindo configurar o número de *sets*, *games* e pontos em cada *game* 😎

Problema 2: Uma Corrida de Revezamento

A *corrida de revezamento* surgiu durante os jogos olímpicos da Grécia e, de maneira geral, é uma competição em que os membros de uma equipe revezam-se na conclusão de partes do local no qual a competição é feita ou realizando todos uma determinada ação. Na modalidade do atletismo, a corrida de revezamento consiste de equipes de corredores, cada uma em uma raia, e esses corredores revezam-se em distâncias definidas carregando um bastão. Ao percorrer a distância estabelecida, o bastão deve ser passado de um atleta para outro e assim sucessivamente, de forma síncrona, até chegar ao término do percurso total.

Nos tempos modernos há dois tipos básicos de corrida de revezamento, 4 x 100 e 4 x 400: a corrida é dividida em quatro etapas, cada uma contendo respectivamente 100 e 400 metros a serem percorridos. No fim de cada etapa, o atleta tem um espaço de 20 metros para entregar o bastão. Esse momento é delicado, pois uma entrega incorreta pode gerar a desclassificação, o que também pode ocorrer caso o bastão não seja entregue nesse espaço de 20 metros. Normalmente o corredor que vai passar o bastão para o próximo dá um sinal sonoro (um grito, por exemplo) para chamar sua atenção para que estenda a mão e esteja pronto para imediatamente pegar o bastão. Dessa forma, a transmissão do bastão requer uma sincronização perfeita para que a etapa não seja perdida, sendo necessário que os dois corredores que estejam envolvidos na transmissão do bastão estejam prontos exatamente ao mesmo tempo.

Escreva um programa em Go que simule uma corrida de revezamento em que uma equipe possui quatro corredores. O segundo, terceiro e quarto corredores não podem começar a correr até que recebam o bastão entregue pelo corredor que o antecedeu. Para simular a corrida do corredor na etapa em questão, pode-se utilizar o método `Sleep` provido pelo pacote `time` por um intervalo de tempo predefinido. O bastão deve ser passado para o próximo corredor até que o último corredor conclua o trecho a ser percorrido, momento em que o programa deverá ser encerrado. Por questões de simplicidade, desconsidere as regras válidas no mundo real em que o bastão necessita ser entregue nos 20 metros finais de cada etapa a ser percorrida.

Extra: Modifique o programa implementado para que haja mais de uma equipe de atletas participando da corrida de revezamento, cada uma em uma raia, de forma similar ao que ocorre no mundo real. Ao término de sua execução, o programa deverá informar qual das equipes venceu a prova.

2 Tarefas

A tarefa central a ser realizada neste trabalho consiste em escolher **um** dos dois problemas descritos anteriormente e implementar uma solução concorrente para ele utilizando a linguagem de programação Go. A implementação deverá garantir corretude do programa com relação a concorrência e aplicar de forma adequada as facilidades de gorotinas. Para promover sincronização entre as gorotinas, poderá ser feito uso de canais de comunicação (com ou sem *buffer*) e/ou de mecanismos de sincronização avançada providos pelo pacote `sync` da linguagem. É **obrigatória** a impressão, na saída padrão, de

mensagens que permitam acompanhar as tarefas que estão sendo realizadas durante a execução do programa.

3 Autoria e política de colaboração

O trabalho deverá ser feito **individualmente**. O trabalho em cooperação entre estudantes da turma é estimulado, sendo aceitável a discussão de ideias e estratégias. Contudo, tal interação não deve ser entendida como permissão para utilização de (parte de) código fonte de colegas, o que pode caracterizar situação de plágio. Trabalhos copiados em todo ou em parte de outros colegas ou da Internet serão sumariamente rejeitados e receberão nota zero.

4 Entrega

O trabalho deverá ser entregue até as **23h59 do dia 2 de fevereiro de 2022**. O código fonte referente à implementação do problema escolhido, disponibilizado sem erros e devidamente testado e **obrigatoriamente comentado**, deverá ser submetido através da opção *Tarefas* da Turma Virtual do SIGAA. Se for o caso, é possível fornecer, no campo *Comentários* do formulário eletrônico de submissão da tarefa, o endereço para um repositório destinado ao controle de versões ou do compartilhamento no Go Playground[2] (https://go.dev/play/), porém esta opção não exclui a necessidade de submissão de arquivo via SIGAA.

5 Avaliação

A avaliação deste trabalho será feita principalmente sobre os seguintes critérios: (i) utilização correta de gorotinas e dos mecanismos de sincronização providos pela linguagem de programação Go; (ii) a corretude da execução do programa implementado, tanto com relação a funcionalidades quanto a concorrência, e; (iii) a aplicação de boas práticas de programação, incluindo legibilidade, organização e documentação de código fonte. Este trabalho possuirá nota máxima de 10,0 (dez) pontos.

[2] Atente para o fato de que a geração de números aleatórios **não funciona adequadamente** no Go Playground em razão de este ser determinístico. Como o ambiente é sempre iniciado com o mesmo instante de tempo, a semente para geração de valores (pseudo)aleatórios será sempre a mesma e consequentemente os valores gerados serão sempre os mesmos. Uma discussão interessante sobre essa e outras questões relacionadas à geração de valores (pseudo)aleatórios em Go pode ser encontrada neste endereço: https://flaviocopes.com/go-random/.