UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
PROGRAMAÇÃO I
TRABALHO COMPUTACIONAL – 2018/1
Profª: Claudia Boeres
Entrega: 29/06/2018 (sexta-feira)

Leia atentamente TODO o enunciado do trabalho (a especificação do problema e os detalhes sobre a confecção, submissão e avaliação do trabalho). Uma leitura inapropriada do enunciado pode ser extremamente danosa a sua nota.

1. Apresentação

Este trabalho computacional deve ser realizado em dupla. Cada dupla deve desenvolver uma solução computacional usando a linguagem Haskell para o problema descrito neste trabalho. É importante que seja usado o método de resolução de problemas apresentado na disciplina de Programação I que inclui as seguintes etapas:

- Compreensão do problema;
- Planejamento do teste da solução;
- Elaboração da solução (preferencialmente, elaboração de várias soluções, de onde possa ser escolhida a melhor);
- Codificação em Haskell da solução;
- Elaboração do Plano de teste para a sua solução incluindo a geração dos dados de entrada;
- Comparação dos resultados obtidos, no ambiente ghci, com os resultados esperados, segundo o plano;
- Avaliação da solução obtida, identificando pontos fortes e fracos;

2. Descrição do problema

Atracação de navios no cais de um porto

O transporte marítimo é uma das atividades principais desenvolvidas na área de comércio internacional. Por conta disso, há um intenso fluxo de navios e contêineres nos portos, fazendo-se necessário forte investimento para acomodação dos navios no cais dos portos, a fim de efetuarem o carregamento de produtos a serem transportados para outros pontos do país e do mundo. Os navios atracam nos berços e permanecem lá durante um período para que o carregamento de produtos possa ser efetuado. Devido ao alto custo relacionado à espera dos navios para atracarem nos berços, o ideal seria diminuir o máximo possível o tempo de atracação e atendimento dos navios. Desta forma, procura-se alocar navios às posições de atraque de forma que se maximize a utilização de espaço do cais, minimizando o tempo de serviço.

No contexto deste trabalho, considere um porto que tem o seu cais dividido em vários berços e somente um navio é atendido por vez em cada berço, independente do seu tamanho. As informações relativas a um navio compreendem o seu número identificador,

a hora de chegada no porto, a hora de partida e a quantidade total de produtos, em toneladas, a ser armazenada no navio. Essas informações são representadas por uma tupla do tipo (Int, Int, Int, Int). Dos berços, é preciso conhecer o seu número identificador e os seus horários de abertura e fechamento, representados pela tupla (Int, Int, Int). Cada navio, quando alocado em um berço do porto, possui um tempo de atendimento, que pode diferir de um berço para outro. Então, também é necessário informar os tempos de atendimento de cada navio em cada berço do porto. Isto é representado por meio de uma lista de tamanho igual ao número de berços do porto, contendo listas de tamanho igual ao total de navios a serem atendidos no porto, contendo por sua vez, os tempos de atendimento de cada navio no berço. O tempo de atendimento nulo indica que o respectivo navio não pode atracar naquele berço. É possível também informar quais navios foram alocados em um berço. Isso é representado por uma dupla, onde o primeiro componente representa o berço e o segundo componente, a lista de navios que atracaram naquele berço.

Apresentamos a seguir um exemplo de todas essas informações. A lista **listaNavios** contém as informações de 5 navios e **listaBercos**, de 2 berços. A lista **infoPorto** contém todos os tempos de atendimento de todos os navios em cada berço. Cada posição dessa lista com dois elementos está relacionada respectivamente com os berços 1 e 2. Por fim, a dupla **naviosAlocadosBerco** informa que os navios 4 e 5 atracaram no berço 1 e os navios 3 e 1, no berço 2.

listaNavios = [(1,5,16,30), (2,6,18,30), (3,3,12,50), (4,4,22,50), (5,11,20,80)]
listaBercos = [(1,4,20), (2,3,18)]
infoPorto = [[1, 6, 4, 4, 6], [2, 0, 1, 0, 5]]
naviosAlocadosBerco1 = ((1,4,20), [(4,4,22,50), (5,11,20,80)])
naviosAlocadosBerco2 = ((2,3,18), [(3,3,12,50), (1,5,16,30)])
naviosAlocadosBerco = [naviosAlocadosBerco1, naviosAlocadosBerco2]

Considerando a entrada de dados no formato descrito acima ilustrado com exemplos, faça um script em Haskell com funções que sejam capazes de resolver os problemas a seguir. Divida o seu script em módulos e informe no cabeçalho de cada módulo, qual a característica do conjunto de funções que pertence àquele módulo. Em negrito, iniciando cada enunciado dos problemas propostos, é sugerido o nome da função que deve ser elaborada.

1. (**atendido**) Dado um navio, um berço e as informações dos tempos de atendimento dos navios nos berços do porto, faça uma função que verifique se o navio pode ser atendido no berço. Exemplo:

> atendido (1,5,16,30) (1,4,20) infoPorto
Resposta: True

2. (**filaNavios**) Considere que vários navios chegam ao porto para serem carregados com produtos para exportação. Como o processo de carregamento pode ser muito demorado, uma fila de navios é formada a medida que eles vão chegando. Dada uma lista de navios que devem atracar no porto, faça uma função que a organize por ordem de chegada dos navios. Exemplo:

> filaNavios listaNavios
Resposta: [(3,3,12,50), (4,4,22,50), (1,5,16,30), (2,6,18,30), (5,11,20,80)]

3. (**tempoOcioso**) Um berço possui uma janela de tempo de trabalho, dentro da qual ele pode atender vários navios, um por vez. Considere um berço e uma lista de navios atendidos nele. Informe o total de tempo ocioso no berço dentro da sua janela de tempo de trabalho, ou seja, informe o tempo total sem atendimento de navios durante a janela de tempo de trabalho do berço. Exemplo:

> tempoOcioso (1, 4, 20) naviosAlocadosBerco1
Resposta: 6

4. (**bercoOcioso**) Dentre todos os berços de um porto, indique o identificador daquele com maior tempo ocioso. Exemplo:

> bercoOcioso bercos naviosAlocadosBerco
Resposta: 2

5. (**naviosCandidatosBerco**) Informe para cada berço, os navios da lista de navios que podem atracar e ser atendidos no berço

> naviosCandidatosBerco bercos listaNavios

Resposta: [(1, [(1,5,16,30), (2,6,18,30), (4,4,22,50), (5,11,20,80)]), (2, [(1,5,16,30), (3,3,12,50), (5,11,20,80)])]

6. (**insereNavioBerco**) Dado um navio e um berço com a lista de navios atendidos nele, verifique se o navio dado como entrada também pode ser atendido neste berço. Além disso, indique na sua resposta, a quantidade total de produtos carregados nos navios que foram alocados naquele berço. Exemplo:

> insereNavioBerco (1,5,16,30) naviosAlocadosBerco1
Resposta:(1, 160, [(4,4,22,50), (1,5,16,30) , (5,11,20,80)])

7. (**esperaNavio**) Dada um navio, indique o tempo de espera do navio caso ele tenha sido alocado no berço. Se não foi alocado, retorne a mensagem: “O navio id_n não foi alocado no berço id_b”, onde id_n e id_b indicam respectivamente os identificadores do navio e do berço. Exemplo:

> esperaNavio (1,5,16,30) naviosAlocadosBerco1
Resposta: 3

8. (**constroiAlocacaoBerco**) Dado um berço, a fila de navios ordenada pelos seus tempos de chegada e seus respectivos tempos de atendimento, informe a lista de todos os navios que podem ser alocados naquele berço. Indique os identificadores dos navios e seus horários de chegada e partida no berço. Além disso, deve-se computar a quantidade total de produtos carregados nos navios que foram alocados naquele berço. Exemplo:

> constroiAlocacaoBerco (1,4,20) (filaNavios listaNavios) infoPorto
Resposta: ((1,4,20), 110, [(4,4,8), (1,8,9), (2,9,15)]

ENTREGA DO TRABALHO

Data de Entrega: O trabalho deverá ser entregue até às **23:59** horas do dia **29/06/2018** (sexta-feira).

Forma de Entrega e observações importantes:

- O trabalho deve ser feito em dupla.
- Devem ser construídos exemplos além do disponível neste enunciado (mínimo de 3 e máximo de 5). Os arquivos de entrada devem ser construídos pela própria dupla, para teste, e devem ser enviados juntos com o código do trabalho.
- O trabalho com erros de sintaxe não será corrigido.
- Trabalhos evidentemente iguais e/ou com plágio receberão nota ZERO.
- **Forma de entrega:** Compactar o script do seu programa Haskell com o nome tc-CC-nome.hs assim como seus módulos em um arquivo compactado nomeado por tc-CC-nome.zip. Enviar por e-mail o arquivo compactado para boeres@inf.ufes.br e thiagoborges533@gmail.com. Tanto no nome do script principal, quanto no nome do arquivo compactado, substitua o termo *nome* pelo primeiro e último nomes dos componentes do grupo. Exemplo: para o grupo formado por Julio Cabral e João Silva, o nome do arquivo será tc-CC-juliocabral-joaosilva.hs. Não usar caracteres especiais tais como ç, ã, õ, é, etc etc no nome do arquivo.
- O **assunto do e-mail enviado** deverá conter a sigla do curso (CC), o nome da disciplina abreviada (ProgI), além do nome de cada componente da dupla, no formado explicado no item anterior, e o semestre letivo. Desta forma, deve seguir o seguinte formato: CC:<disciplina>:<nome1>:<nome2>:20181. Exemplo: para o grupo formado por Julio Cabral e João Silva, o assunto do e-mail deve ser CC:ProgI:juliocabral-joaosilva:20181.
- **Faça o cabeçalho do seu script:** Coloque como comentário no topo do seu script, os nomes dos participantes da dupla, a data e o título do trabalho em todos os scripts.

- Avaliação individual relativa ao projeto de programação pode ser realizada através de entrevista ou teste (a ser decidido posteriormente). No caso de entrevista, será avaliada a fluência de cada estudante quanto ao processo de desenvolvimento do projeto de programação, principalmente quanto ao que aprendeu, como foi construída a solução e a participação nas atividades realizadas pelo grupo, através de entrevista a ser marcada. A nota obtida na entrevista é multiplicada pela nota atribuída ao código. No caso de teste, uma questão sobre o trabalho será aplicada no laboratório e um peso associado a essa questão será multiplicado pela nota do trabalho.

- O recebimento dos trabalhos é automatizado. Siga as instruções à risca pois algum erro na submissão pode inviabilizar a entrega do seu trabalho. Não deixe para enviar

seu trabalho nos momentos finais de seu prazo. É comum a ocorrência de problemas em virtude de erros na submissão. Logo, enviem com algumas horas de antecedência para que haja tempo hábil para eventuais correções.

BOM TRABALHO!!!