1ª Prova de EDA-II - Curso: Engenharia de Software

Data: 3 de Outubro de 2023 **Horário limite:** 12:00

(50 pontos) 1) Considere a implementação de uma tabela hash utilizando DOUBLE HASH, para tratamento de colisões tendo 10 como o limite de colisões aceitável. Responda as perguntas abaixo:

a) (30 pontos) O `grau` de colisão é a quantidade de colisões que um hash específico possui. Implemente uma função `grau` que recebe a tabela hash, e uma chave, a função deve devolver a quantidade de colisões para esta hash.
   - É possível responder o `grau` de uma hash qualquer em tempo constante, i.e $\mathcal{O}(1)$?

b) (20 pontos) Qual o impacto nas operações(inserção,remoção,busca) quando se aumenta o limite de 10 colisões para $N/2$, sendo N a quantidade de Itens que podem ser armazenados na tabela hash?

(20 pontos) 2) Insira em uma árvore Red Black Descendente Esquerdista, inicialmente vazia, os seguintes caracteres: `M E N I N A S C O M P`

- Mostre a árvore `2-3` equivalente. **+2 pontos adicionais**

(40 pontos) 3) (Checkpoint de corrida) Suponha que você tenha sido contratado para escrever um programa que identifica fraudes em corridas. A ideia é que todos os corredores passem por *checkpoints* espalhados no caminho. Sempre que um corredor passar por este checkpoint será fornecido ao seu programa o horário (em segundos desde 1 de janeiro de 1970 00:00), o código do corredor e em qual *checkpoint* ele está passando. Os *checkpoints* possuem números incrementais, $1, 2, 3, 4...N$. Você deverá mostrar um aviso na tela sempre que identificar que um corredor pulou um *checkpoint*. Ao final da corrida, quando a entrada terminar em `EOF`, o seu programa deveria imprimir a lista de todos os corredores que passaram por todos os checkpoints, a impressão deve ser ordenada pelo código do corredor. Uma vez que um corredor "pula" um *checkpoint* ele nunca volta atrás e continua correndo em direção aos *checkpoints* de maior número.

Com base neste problema, seguem as seguintes afirmativas:

- Não sabemos a quantidade de números que serão lidos;
- Não sabemos a quantidade de corredores, nem de *checkpoints*;
  - é possível inferir a quantidade de checkpoints quando termina a corrida, pois pelo menos um corredor passa em todos os *checkpoints*
  - é possível inferir a quantidade de corredores quando termina a corrida, pois todos os corredores passam por pelo menos um *checkpoint*
- Você $N\tilde{A}O$ deve implementar um monte de código neste problema. Indique o raciocício e os custos da operações envolvidas
- Qual estratégia você utilizará para resolver este problema?
  - Qual será a estrutura de dados? Você precisará fazer algum tratamento na informação?
  - Lembre que você poderá ler a entrada apenas uma vez, e ela é construída aos poucos durante a corrida
- Qual o espaço em memória adicional necessário para resolver o problema?
- Mostre a estrutura de dados utilizada e os custos envolvidos nas operações de armazenamento e impressão dos elementos pedidos no problema;
- Qual é o pior tipo de entrada que pode ser fornecido para o seu programa? Há algum impacto nas operações a ordem em que os dados entram?
- Apresente um protótipo das principais funções da sua solução;
- A sua solução deve ser a mais eficiente possível, e deve utilizar alguma estrutura aprendida neste semestre

Exemplo de entrada:

```
1626228376 1001 1
1626228482 1002 1
1626228579 1001 2
1626228580 1002 3
1626228615 1002 4
1626228710 1001 3
EOF
```

Exemplo de saída:

```
Aviso: Corredor 1001 pulou o checkpoint 2

Corredores que passaram por todos os checkpoints:
1002
```