Descrição: Implementar em Java um algoritmo paralelo/concorrente para gerenciamento dinâmico de memória.

RF - Requisito Funcional
RNF - Requisito Não Funcional

[RF-01] - Gerador de Requisições:
Deve gerar requisições aleatórias de alocação de memória, assim simulará a criação de demandas para alocação de variáveis (espaços) na memória. O número total de requisições deve ser definido pelo usuário para testes de carga (workload).
Cada requisição deve ter/especificar:

- ID único (deve ser escrito em todas as posições da heap ocupadas pela variável correspondente ao ID)
- Tamanho aleatório em bytes, em que o usuário defina o valor mínimo e máximo que a requisição pode ter.
- Quantidade de páginas utilizadas e lista para identica-las, importante para desalocação e desmapeamento.

[RF-02] - Gerenciamento da Heap:
Implementar a heap como um vetor de inteiros (inteiro = 4 bytes), ela armazena o ID de uma requisição para simular alocação. Essa será a memória dinâmica e núcleo do projeto. O sistema enxerga a heap dividida em frames e aloca requisições como multiplos da página. O tamanho da heap em KB deve ser definido pelo usuário.

[RF-03] - Alocação de Memória:
O sistema deve utilizar paginação para gerenciar a alocação de memória. O tamanho da pagina (frame) em bytes deve ser definido pelo usuário. Ao alocar uma requisição, deve ser calculado número de páginas necessárias, arredondando para cima e admitindo fragmentação interna. Verificar se há frames (espaço na heap) livres, se houver, consequentemente há páginas livres. Precisa identificar quais são as paginas e então alocar e atualizar a tabela de páginas. Caso não haja frames livres, deve acionar o algoritmo de liberação.

[RF-04] - Algoritmo de Liberação de Memória:
Liberar pelo menos 30% do tamanho da heap. A requisição deve ser inteiramente liberada. A quantidade de requisições liberadas depende da quantidade de páginas da requisição e quantidade de páginas dos 30% da heap. Usar política FIFO para liberação de memória. Manter registro das variáveis alocadas em ordem de chegada para implementar FIFO.

[RF-05] - Relatório:
Ao final da execução, o sistema deve exibir:

- Número total de requisições atendidas (alocadas na heap)
- Tamanho médio das variáveis alocadas
- Número total de variáveis removidas da heap
- Tempo total de execução.

[RNF-01] - Desempenho:
O desempenho deve ser avaliado executando testes com diferentes tamanhos de heap, página e números de requisições. O sistema deve ser capaz de lidar com grande volume de requisições. A implementação paralela deve mostrar ganhos de desempenho em relação à versão sequencial

[RNF-02] - Manutenibilidade:
O código deve ser modular e bem documentado para facilitar futuras atualizações e manutenção. O uso de boas práticas de programação deve ser priorizado.

[RNF-03] - Linguagem de Programação:
O sistema deve ser desenvolvido utilizando a linguagem de programação Java.