Lista de Exercícios 2 - Estruturas de Dados Lineares

Oficina de AEDs2 - Prof. Matheus Pereira

1. Exercício 1 (em C)

Contexto: Um sistema embarcado crítico para controle de uma fábrica automotiva precisa gerenciar três tipos de tarefas com diferentes prioridades e comportamentos:

1. **Tarefas de Tempo Real (Alta Prioridade)**: Devem ser processadas imediatamente (sistema de airbags, freios)
2. **Tarefas Periódicas (Média Prioridade)**: Executadas em intervalos regulares (monitoramento de sensores)
3. **Tarefas de Background (Baixa Prioridade)**: Processadas quando o sistema está ocioso (log de dados)

Enunciado: Implemente um sistema integrado que utilize **três estruturas de dados diferentes** para gerenciar esses tipos de tarefas (implemente tratamento de overflow/underflow para todas as estruturas):

Estruturas Requeridas:

1. **Pilha Estática** (Para tarefas de tempo real)
   - Implemente com array de tamanho fixo
   - Operações: `push_emergency()`, `pop_emergency()`
2. **Fila Circular** (Para tarefas periódicas)
   - Implemente com array e índices circulares
   - Operações: `enqueue_periodic()`, `dequeue_periodic()`
3. **Lista Ordenada** (Para tarefas de background)
   - Mantenha ordenada por prioridade (0 = mais alta, 10 = mais baixa)
   - Operações: `insert_background()`, `remove_background()`, `get_highest_priority()`

Observação: seu código não está restrito somente às funções sugeridas!

Requisitos do Sistema: Utilize a estrutura abaixo para definir as tarefas simuladas.

```c
#define MAX_EMERGENCY 10
#define MAX_PERIODIC 20
#define MAX_BACKGROUND 30

typedef struct {
    int task_id;
    int priority; // 0-10 (0 = maxima prioridade)
} Task;
```

Funcionalidades Essenciais para Implementar:

1. **Scheduler Integrado**:

   ```c
   Task get_next_task();
   ```

   Que retorna a próxima tarefa a ser executada seguindo as regras:

   - Sempre verifica primeiro a pilha de emergência
   - Depois a fila circular de tarefas periódicas
   - Finalmente a lista de tarefas de background

2. **Mecanismo de Promoção**:

   ```c
   void promote_to_emergency(int task_id);
   ```

   Que move uma tarefa da lista de background para a pilha de emergência

3. **Estatísticas do Sistema**:

   ```c
   void print_system_stats();
   ```

   Que mostra o estado atual de todas as estruturas

Análise de Complexidade Requerida: Para cada operação implementada, analise e justifique a complexidade de tempo das operações individuais de cada estrutura.

Observação: Mantenha a ordenação eficiente na lista de background.