**Aluno:Guilherme Lucas Rodrigues Cesar**

**Matéria: Qualidade de Software e Governança**

**Curso: Análise e Desenvolvimento de Sistemas**

**Segundo Semestre- Matutino- Ceilândia**

**Relatório de Refatoração e Análise de Código – Pilha em C**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Mudanças Realizadas:**

1. **Verificação de Erro no scanf**:
   * Agora o scanf é verificado com if (scanf("%99s", expressao) != 1), garantindo que o programa lida corretamente com falhas de leitura de entrada.
2. **Acesso Seguro à Pilha em comparar**:
   * Adicionada a verificação de topo == NULL antes de acessar o topo da pilha, prevenindo o erro de acesso à memória inválida.
3. **Checagem de Pilha Vazia**:
   * A pilha é verificada corretamente após a iteração pela expressão para garantir que não restem elementos não fechados. Se a pilha não estiver vazia, a expressão será considerada inválida.
4. **Tratamento de Erros de Alocação**:
   * O código de erro ao falhar na alocação de memória com malloc foi mantido com exit(1), mas com mais clareza. Dependendo do contexto, isso poderia ser substituído por um código de erro mais detalhado.
5. **Comentários Explicativos**:
   * Comentários foram adicionados para explicar cada parte importante do código, facilitando o entendimento e manutenção.

**Problemas no Código com Erros:**

1. **malloc sem Verificação Completa**:
   * A alocação de memória é feita sem um tratamento de erro robusto. O código apenas imprime uma mensagem e chama exit(1), mas não há um tratamento mais adequado para erros de alocação de memória.
2. **Acesso a Ponteiro Nulo em comparar**:
   * Na função comparar, o código assume que a pilha nunca estará vazia quando for chamada. No entanto, se a pilha for vazia, isso causaria um acesso inválido à memória. O código não verifica isso adequadamente.
3. **Verificação Incompleta de pilha no Final**:
   * A verificação int resultado = (pilha == NULL); está correta no conceito, mas não leva em consideração a possibilidade de a pilha ter elementos extras após a validação da expressão.
4. **scanf sem Verificação de Erro**:
   * O scanf não verifica se a entrada foi lida corretamente. Isso pode levar a erros se o usuário não inserir uma string válida.
5. **Falta de Comentários e Explicações**:
   * O código não tem comentários explicativos, o que pode dificultar a compreensão e a manutenção.