**Universidade Federal de Uberlândia**
**Curso de Graduação em Gestão da Informação**
**4º Trabalho de Estruturas de Dados I – Prof. Daniel A. Furtado**
**Trabalho Individual – Listas Duplamente Ligadas e Circular**

## INTRUÇÕES GERAIS – LER ATENTAMENTE

- Esta atividade deve ser realizada individualmente;
- Este trabalho deve ser feito utilizando a linguagem C# ou Python. Trabalhos implementados utilizando outras linguagens serão anulados;
- **Todo o código deve ser colocado em um único arquivo**, pois apenas um arquivo (.cs ou .py) deve ser entregue. Não crie arquivos adicionais para as classes (o único arquivo a ser entregue deve conter todas as definições de classes, assim como o programa principal).
- **Não compactar** os arquivos no momento da entrega;
- Os recursos adequados das linguagens devem ser utilizados;
- Esteja atento às **observações sobre plágio** apresentadas no final desde documento;
- Trabalhos com implementações utilizando trechos de códigos retirados de sites da Internet, de trabalhos de colegas de classe ou de semestres anteriores serão anulados;
- As implementações não devem conter qualquer conteúdo de caráter imoral, desrespeitoso, pornográfico, discurso de ódio, desacato, etc.;
- O trabalho deve ser entregue até a data definida pelo professor em aula síncrona pelo Sistema de Aplicação de Testes (SAAT);
- Trabalhos enviados por e-mail ou por outros meios digitais (como o MS Teams) **não serão considerados** (veja instruções no final);

## Parte 1

Implementar o TAD **Lista Duplamente Ligada Circular** utilizando as definições de classe e métodos apresentadas nos slides de aula "Módulo 5 – Listas Circulares e Duplamente Ligadas". Devem ser implementadas as operações listadas a seguir:

- **AddFirst – adiciona um elemento no início da lista**
- **AddLast – adiciona um elemento no final da lista**
- **InsertAt(i, item) – insere o elemento item na posição i da lista**
- **RemoveFirst – remove o primeiro elemento da lista**
- **RemoveLast – remove o último elemento da lista**
- **Remove(item) – remove a primeira ocorrência do elemento item**

- **Get(i) – retorna o elemento na posição i da lista, sem remover**
- **Set(i , newItem) – substitui o item na posição i da lista por newItem**
- **Find(item) – retorna a posição da primeira ocorrência do item (ou -1)**
- **DisplayFirstToLast() – exibe os itens da lista na sequência natural**
- **DisplayLastToFirst() – exibe os itens da lista na sequência inversa**

Em seguida, crie um código principal para exemplificar o uso da lista duplamente ligada circular. O código deve ilustrar o emprego de todos os métodos implementados.

### Parte 2 – Problema de Josephus

Considere que haja N pessoas formando um círculo aguardando para serem executadas. O processo de execução é guiado pela lógica a seguir, de tal forma que apenas a última pessoa a permanecer no círculo terá a vida poupada.

Escolhe-se inicialmente um número M e um ponto de partida no círculo. Em seguida, M pessoas são contadas no sentido anti-horário e a pessoa onde a contagem termina é então escolhida para ser executada (depois de executada, a pessoa é removida do círculo). O mesmo processo se repete, recomeçando a contagem de M sempre pela pessoa à direta da última que foi executada.

Implementar um método de nome `CountNodesAndRemove(int m)` para viabilizar a simulação do problema de Josphus utilizando a lista circular implementada na primeira parte deste trabalho. A cada vez que for chamado, o método deve utilizar o parâmetro `m` para percorrer a lista a partir do último ponto de parada, saltar `m` nós e remover o nó seguinte. Considere como ponto de partida, para a primeira chamada do método, o início da lista (**dica**: pode ser necessário utilizar atributos adicionais privados para salvar o local da última remoção, pois este será o ponto de início da contagem quando o método for chamado novamente).

Em seguida, crie um código principal para exemplificar o uso do método.

**OBS**: Ao término do trabalho o programa principal deve conter todos os trechos de código de teste, na sequência das partes do exercício.

## Entrega

Um único arquivo (.cs ou .py) contendo todas as definições de classes e também o programa principal deve ser enviado pelo Sistema de Aplicação de Testes (SAAT) até a data limite indicada pelo professor. Não envie arquivos compactados ou outros arquivos do Visual Studio que não sejam o arquivo de código principal.

## Sobre Eventuais Plágios

Este é um trabalho individual. Os alunos envolvidos em qualquer tipo de plágio, total ou parcial, seja entre equipes ou de trabalhos de semestres anteriores ou de materiais disponíveis na Internet (exceto os materiais de aula disponibilizados pelo professor), serão duramente penalizados (art. 196 do Regimento Geral da UFU). Todos os alunos envolvidos terão seus **trabalhos anulados** e receberão **nota zero**.