Universidade Federal de Uberlândia
Curso de Graduação em Gestão da Informação
2º Avaliação de Estruturas de Dados I – Prof. Daniel A. Furtado
Filas e Listas – Valor: 25 pontos

Data/Hora de Início: 14/09/2021 – **10h40**
Data/Hora de Término: 14/09/2021 – **12h20**

INTRUÇÕES GERAIS

- Esta avaliação deve ser realizada individualmente;
- As soluções devem ser implementadas utilizando a linguagem C# ou Python. Questões com respostas implementadas em outras linguagens não serão consideradas;
- O arquivo correspondente à solução de cada questão (.cs ou .py) deve ser enviado pelo sistema SAAT (www.furtado.prof.ufu.br), sem compactar, **até o horário de término indicado acima**. Apenas **um arquivo deve ser enviado por questão.** Portanto, todo o código da questão (definições de classes, programa principal (se houver), etc.) deve ser colocado no mesmo arquivo.
- Questões não enviadas até a data/hora limite serão automaticamente perdidas pelo aluno (com respectiva nota 0) – **não deixe para enviar os arquivos nos instantes finais!**
- Esta prova pode ser realizada com consulta **apenas** aos materiais disponibilizados pelo professor no endereço www.daniel.prof.ufu.br e no MS Teams, incluindo slides de aula e códigos de exemplo. Porém, NÃO É PERMITIDA a consulta a outros materiais, websites, trabalhos de semestres anteriores, etc. O uso de qualquer material desse tipo será interpretado como plágio e resultará na anulação da prova;
- Todo o código deve ser **indentado corretamente**. Código sem a devida indentação será considerado ilegível, de difícil manutenção, e resultará em **penalização de 30%** sobre o valor da questão;
- A eficiência, clareza e coerência do código também serão avaliadas;
- As implementações não devem conter qualquer conteúdo de caráter imoral, desrespeitoso, pornográfico, discurso de ódio, desacato, etc.;
- Respostas de questões enviadas por e-mail ou pelo Teams **não serão consideradas**;
- **Alunos envolvidos em qualquer tipo de plágio, total ou parcial, terão suas provas anuladas e receberão nota zero (art. 196 do Regimento Geral da UFU).**

Questão 1 (8 pontos)

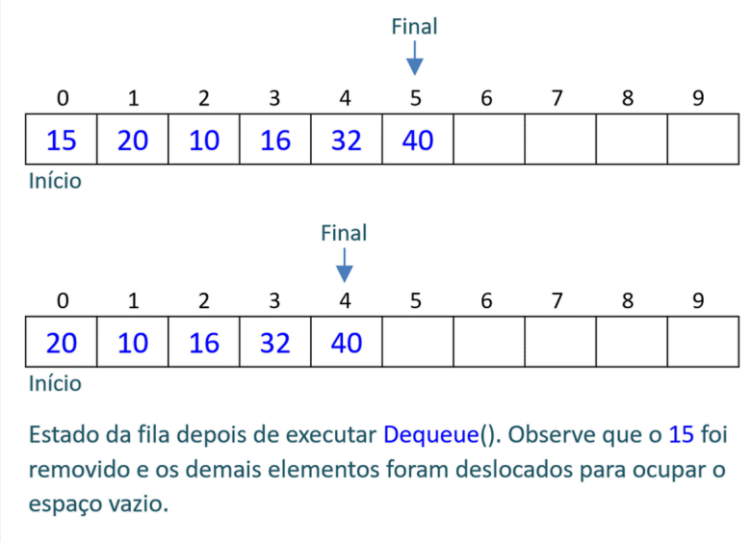
O código a seguir corresponde a uma implementação do Tipo Abstrato de Dados **Fila** utilizando *array*. Nesta implementação, os elementos são inseridos na fila a partir da posição 0 do *array*. A operação Enqueue() insere um novo elemento na próxima posição 'vazia' do *array* utilizando o atributo 'final', conforme apresentado no código.

Insira o código adequado para o método Dequeue(), de acordo às especificações a seguir:

- Dequeue() deve remover e retornar o elemento no início da fila. Dequeue() deve ser implementado de tal forma que, após a remoção, os elementos seguintes devem ser movidos uma posição à esquerda para ocupar o lugar 'vazio' deixado na posição 0 do *array* após a retirada do elemento no início da fila.

- A implementação do método Dequeue() deve ser feita sem criar métodos ou atributos adicionais. A implementação NÃO deve utilizar um segundo ponteiro no início da fila, pois o início da fila será sempre o início do *array*;
- Não modifique nenhuma outra parte do código fornecido;
- Recomenda-se criar um programa principal para testar o método implementado. Porém o programa principal não precisa ser entregue e não será avaliado.

Observe o diagrama a seguir:



Estado da fila depois de executar Dequeue(). Observe que o 15 foi removido e os demais elementos foram deslocados para ocupar o espaço vazio.

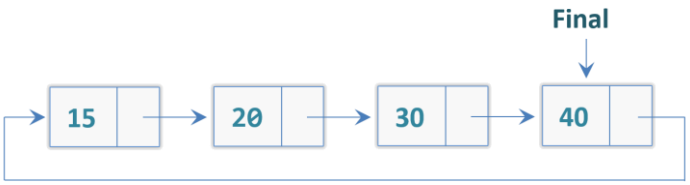
```
public class Queue
{
   // Não acrescente outros atributos
   int[] items;
   int final;

   public Queue(int capacity)
   {
      this.items = new int[capacity];
      this.final = -1;
   }
   public void Enqueue(int newItem)
   {
      if (final == items.Length - 1)
         return;
      final++;
      items[final] = newItem;
   }
   public bool IsEmpty()
   {
      return (final == -1);
   }
   public int Dequeue()
   {
      // preencha com o código adequado
   }
}
```

```
class Queue:

    def __init__(self, size):
        self.items = array.array('i',[0
for i in range(size)])
        self.final = -1

    def Enqueue(self, newItem):
       if (self.final == (len(self.items)
- 1):
          return

       self.final += 1
       self.items[final] = newItem

    def IsEmpty(self):
        return (self.final == -1)

    def Dequeue(self):
        # preencha com o código adequado
```

Atenção: ao terminar a questão anterior, envie o arquivo pelo sistema SAAT imediatamente!

Questão 2 (8 pontos)

O código a seguir é uma implementação parcial de uma lista simplesmente ligada circular com referência para o nó final, conforme ilustrado na figura:



Considere que os elementos na lista estejam **ordenados** e utilize o código fornecido como base para implementar o método `Find(item)`. O método deve buscar pela primeira ocorrência do item na lista e retornar a sua posição, caso esteja presente. Caso contrário (se o item não estiver na lista), o método deve retornar -1. Considere que o primeiro nó tenha posição 0, que o segundo tenha posição 1, etc. Considere também as seguintes restrições:

- Não crie métodos nem atributos adicionais (causará a anulação da questão);
- Não modifique nenhuma outra parte do código fornecido;
- O método deve ser otimizado considerando que a lista esteja ordenada;
- Um programa principal é recomendado para fins de teste, mas não será avaliado.

```
public class Node
{
   public int item;
   public Node next;
   public Node(int newItem)
   {
      this.item = newItem;
   }
}
public class ListaCirc
{
   // Não acrescente outros atributos
   int count;
   Node final;

   public ListaCirc()
   {
      this.final = null;
      this.count = 0;
   }

   public void AddFirst(int newItem)
   {
      Node no = new Node(newItem);
      if (final == null) {
         no.next = no;
         final = no;
      }
      else {
         no.next = final.next;
         final.next = no;
      }
      count++;
   }

   public bool IsEmpty()
   {
      return (final == null);
   }

   public int Find(int item)
   {
      // preencha com o código adequado
   }
}
```

```
class Node:
   def __init__(self, newItem):
      self.next = None
      self.item = newItem

class ListaCirc:
   def __init__(self):
      self.final = None
      self.count = 0

   def AddFirst(self, newItem):
      no = Node(newItem)
      if (self.final == None):
         no.next = no
         self.final = no
      else:
         no.next = self.final.next
         self.final.next = no

      self.count += 1

   def IsEmpty(self):
      return (self.final == None)


   def Find(item):
      # preencha com o código adequado
```

Atenção: ao terminar a questão anterior, envie o arquivo pelo sistema SAAT imediatamente!

Questão 3 (9 pontos)

O código na página a seguir é uma implementação parcial de uma lista duplamente ligada circular. Utilize esse código como base e implemente o método `RemoveAt(p)` para remover o elemento da lista na posição `p`, atendendo às especificações a seguir:

- Não crie métodos nem atributos adicionais (causará a anulação da questão);
- **Não crie** um método "RemoveNode";
- Não modifique nenhuma outra parte do código fornecido;
- Não há necessidade de fazer otimizações para percorrer a lista no sentido horário ou anti-horário, dependendo do valor `p` (com o intuito de reduzir os passos pela metade);
- Recomenda-se criar um programa principal para testar o método implementado. Porém o programa principal não precisa ser entregue e não será avaliado.

```java
public class Node
{
    public int item;
    public Node next;
    public Node prev;

    public Node(int newItem)
    {
        this.item = newItem;
    }
}

public class ListaDuplaCirc
{
    // Não acrescente outros atributos
    int count;
    Node inicio;

    public ListaDuplaCirc()
    {
        this.inicio = null;
        this.count = 0;
    }

    public void AddLast(int newItem)
    {
        Node no = new Node(newItem);
        if (inicio == null) {
            no.prev = no;
            no.next = no;
            inicio = no;
        }
        else {
            no.prev = inicio.prev;
            no.next = inicio;
            inicio.prev.next = no;
            inicio.prev = no;
        }
        count++;
    }

    public bool IsEmpty()
    {
        return (inicio == null);
    }

    public void RemoveAt(int p)
    {
        // preencha com o código adequado
    }
}
```

```python
class Node:
    def __init__(self, newItem):
        self.next = None
        self.prev = None
        self.item = newItem

class ListaDuplaCirc:
    def __init__(self):
        self.inicio = None
        self.count = 0

    def AddLast(self, newItem):
        no = Node(newItem)
        if (self.inicio == None):
            no.prev = no
            no.next = no
            self.inicio = no
        else:
            no.prev = self.inicio.prev
            no.next = self.inicio
            self.inicio.prev.next = no
            self.inicio.prev = no

        self.count += 1

    def IsEmpty(self):
        return (self.inicio == None)


    def RemoveAt(p):
        # preencha com o código adequado
```

Atenção: ao terminar a questão anterior, envie o arquivo pelo sistema SAAT imediatamente!