UNIVERSIDADE
DO ESTADO DE MINAS GERAIS | UEMG
UNIDADE PASSOS

Estruturas de Dados 3
Lista Encadeada Circular

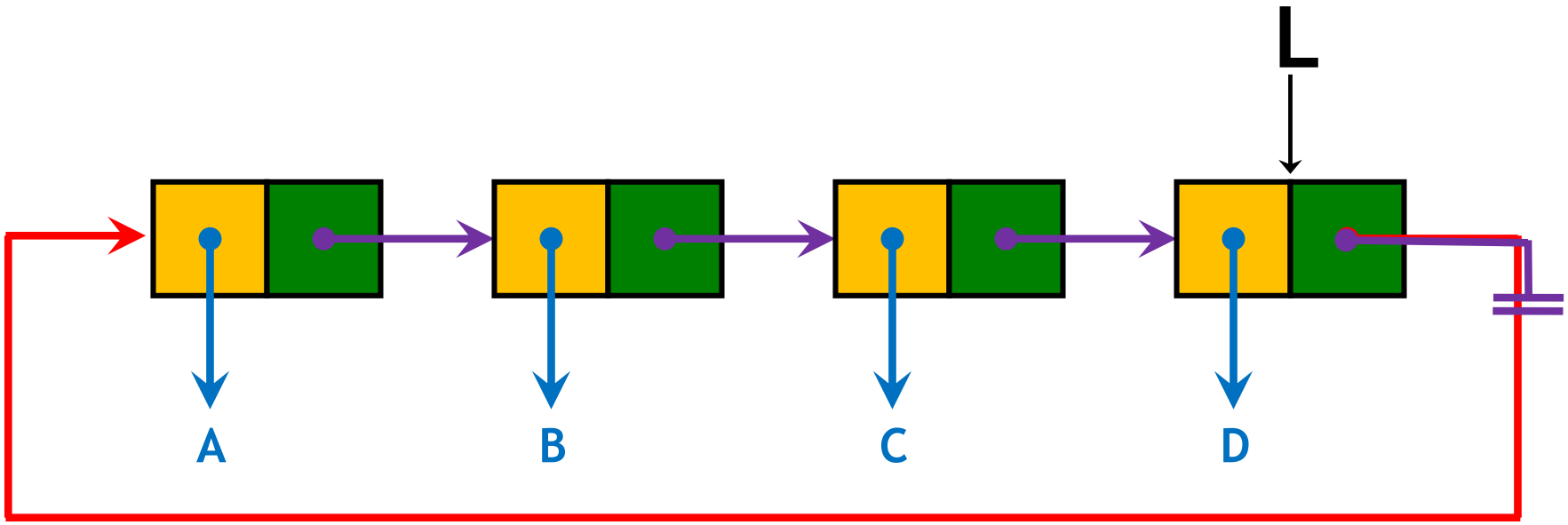
Mailson de Queiroz Proença

Listas Encadeadas Circulares

- Em uma lista circular a última célula da lista aponta para a primeira, formando um ciclo.

- A lista pode ser representada por um ponteiro para um elemento inicial qualquer da lista
  - Neste tipo de lista é possível acessar qualquer célula a partir de qualquer ponto, pois na verdade poderíamos considerar qualquer célula como sendo a primeira.

Listas Encadeadas Circulares

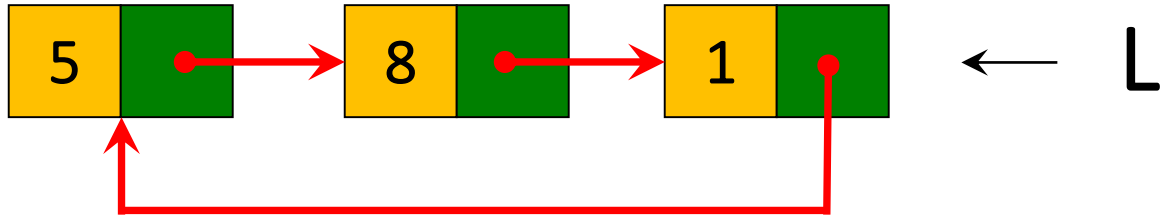


- Não existe mais uma “primeira” e “última” célula natural;
- Por convenção: A **lista aponta** para a **última Célula** e a Célula seguinte torna-se a primeira célula.
  - Última célula: **lista**
  - Primeira célula: **lista->proxima**

Listas Encadeadas Circulares

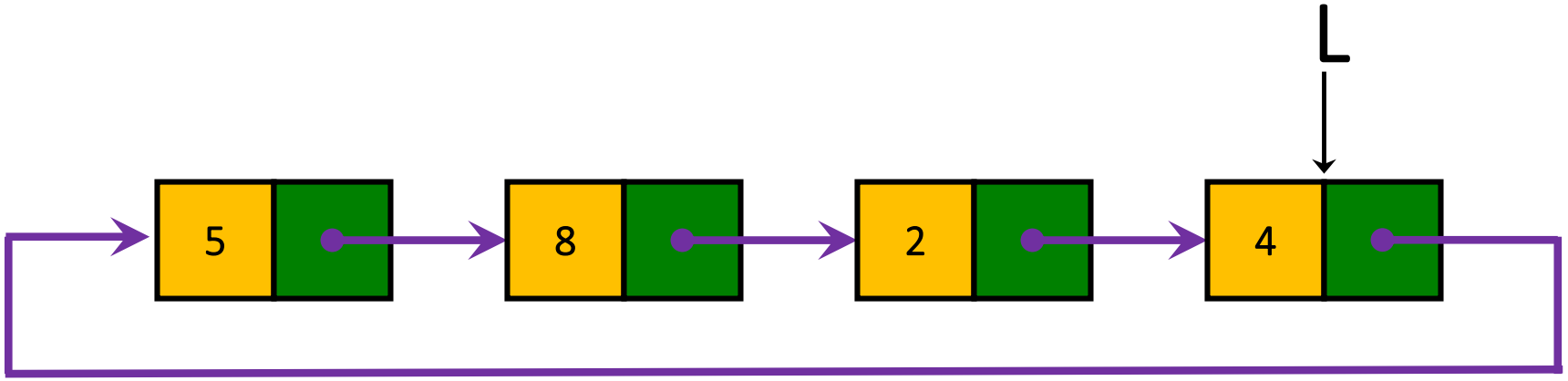
- As operações em uma lista circular são semelhantes as operações em uma lista comum, devemos apenas ter cuidado nas operações que envolvem a célula inicial e a célula final.
- Algumas Operações em Listas Encadeadas Circulares:
  - Percorrer a Lista Circular
  - Inserir Célula no início da Lista Circular
  - Inserir Célula no final da Lista Circular
  - Remover Célula da Lista Circular
  - Esvaziar a Lista Circular
  - Inserir Célula ordenada na Lista Circular

Percorrer uma Lista Circular




```
void ImprimeLista(celula *lista){
      if (lista != NULL){
            //Aponta para a1ª célula da Lista
            celula *aux = lista->proxima;
            while (aux != lista){
                  cout << aux->elem<<endl;
                  aux = aux->proxima;
            }
            cout << lista->elem;
      }
}
```

Inserir um Célula no Início de uma Lista Circular



1. Aloca espaço para o nova Célula
2. Define os valores dos elementos da Célula
3. nova Célula aponta para a primeira Célula da Lista
4. A última Célula aponta para a nova Célula

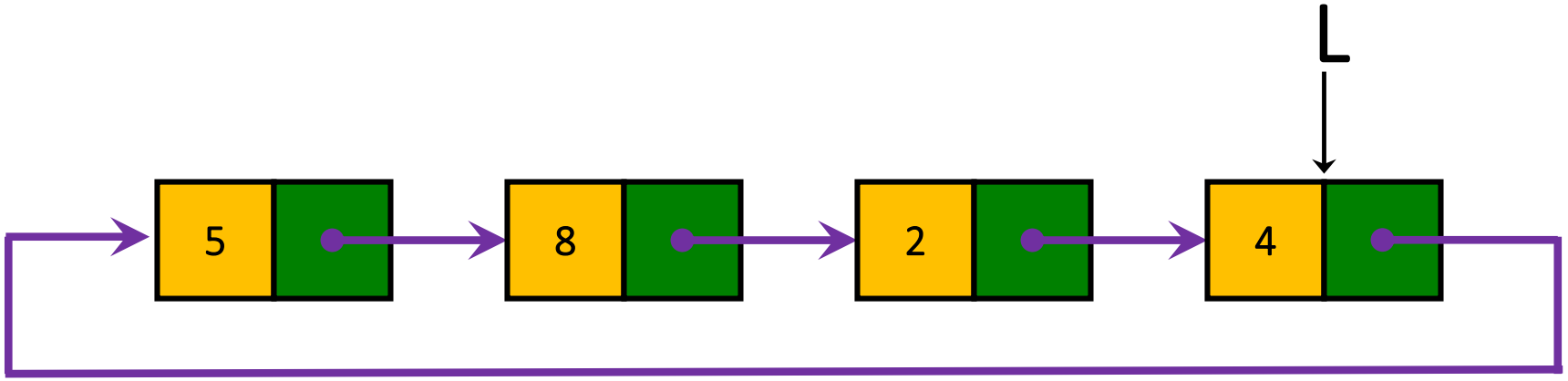
Inserir um Célula no Início de uma Lista Circular

```
celula *InsereNoInicio(celula *lista, int val){
      celula *nova = new celula;
      nova->elem = val;
      if (lista == NULL){
            nova->proxima = nova;
            lista = nova;
            return lista;
      }else{
            nova->proxima = lista->proxima;
            lista->proxima = nova;
            return lista;
      }
}
```

Testando a Lista...

```
int main(){
    celula *lista; //declara lista não inicializada
    lista = CriarLista(); //cria e inicia lista vazia
    lista = InsereNoInicio(lista, 23); //insere o nro 23
    lista = InsereNoInicio(lista, 45); //insere o nro 45
    lista = InsereNoInicio(lista, 18); //insere o nro 18
    ImprimeLista(lista);
    return 0;
}
```

Inserir um Célula no Final de uma Lista Circular



1. Aloca espaço para a nova Célula
2. Define os valores dos elementos da Célula
3. nova Célula aponta para a primeira Célula da Lista
4. A última Célula aponta para a nova Célula
5. A Lista aponta para a nova Célula

Inserir um Célula no Final de uma Lista Circular

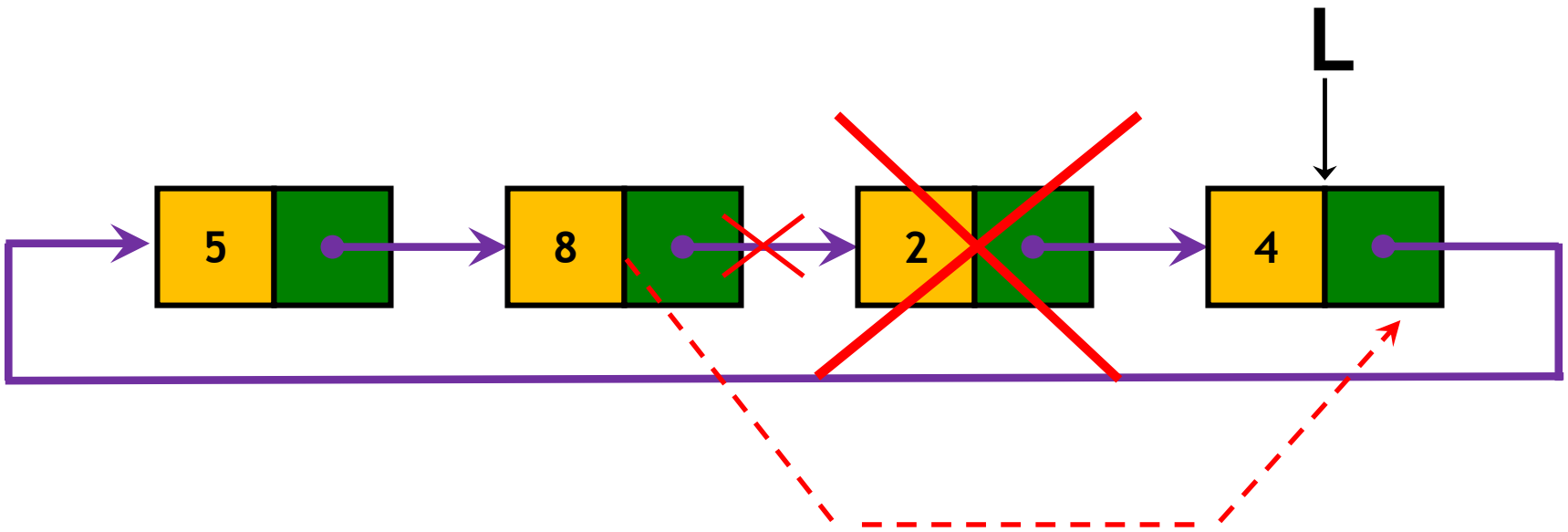
```
celula *InsereNoFim(celula *lista, int val){
      celula *nova = new celula;
      nova->elem = val;
      if (lista == NULL){
            nova->proxima = nova;
            lista = nova;
            return lista;
      }else{
            nova->proxima = lista->proxima;
            lista->proxima = nova;
            lista = nova;
            return lista;
      }
}
```

Testando a Lista...

```c
int main(){
    // insere o elemento 34
    lista = InsereNoFim(lista, 34);
    ImprimeLista(lista);
    // insere o elemento 96
    lista = InsereNoFim(lista, 96);
    ImprimeLista(lista);
    return 0;
}
```

Remover um Célula de uma Lista Circular

1. Percorre a lista procurando a Célula que será removida
2. Remove a Célula



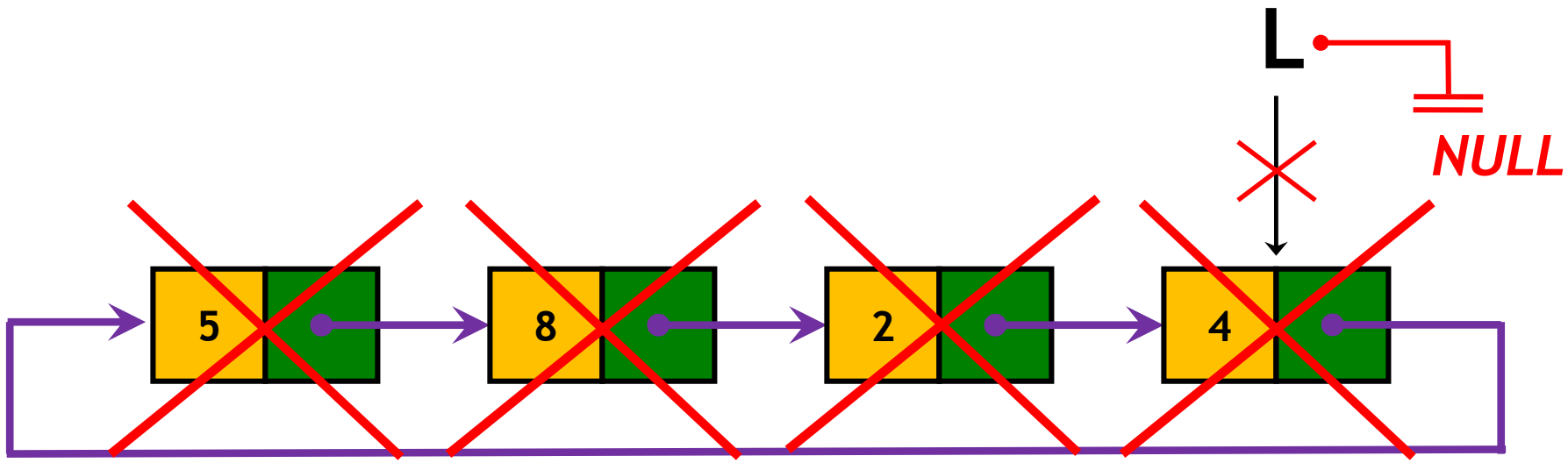
Remover um Célula de uma Lista Circular

```
celula *remover(celula *lista, int valor)
{
            if(lista != NULL)
            {
                        celula *aux = lista->proxima;
                        celula *ant = NULL;
                        if(aux != lista) //verifica se a lista tem mais de uma célula
                        {
                                    while(aux != lista && aux->elemento != valor)
                                    {
                                                ant = aux;
                                                aux = aux->proxima;
                                    }
                                    if(aux == lista && aux->elemento != valor)
                                    {
                                                cout << "Nao encontrou\n";
                                                return lista;
                                    }
                                    else if(aux == lista->proxima) //remove o 1º Célula
                                                lista->proxima = aux->proxima;
                                    else if(aux == lista)// remove o último Célula
                                    {
                                                ant->proxima = aux->proxima;
                                                return ant;
                                    }
                                    else // remove um Célula do meio da lista
                                                ant->proxima = aux->proxima;
                                    delete aux;//libera memória
                        }
                        else if(valor == aux->elemento)
                        {
                                    lista = NULL;
                                    delete aux;//libera memória
                        }
            }
            return lista;
}
```

Testando a Lista...

```
int main(){
      // Remove a Célula 34
   lista = RemoveCelula(lista, 34);
   ImprimeLista(lista);
   // Remove a Célula 96
   lista = RemoveCelula(lista, 96);
   ImprimeLista(lista);
   return 0;
}
```

Esvaziar uma Lista Circular



1. Percorre cada Célula da Lista
2. Remove a Célula (Libera memória)

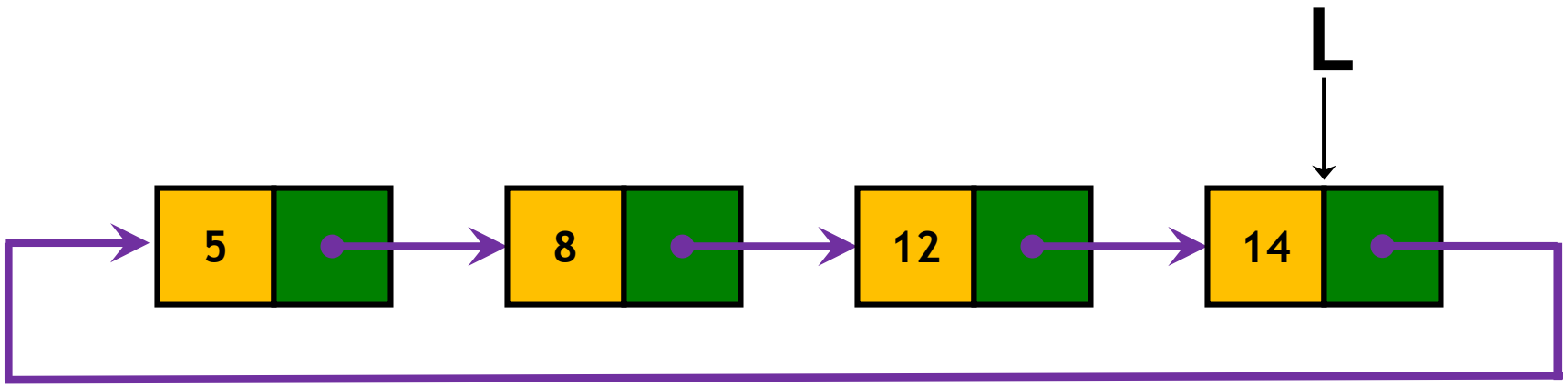
Esvaziar uma Lista Circular

```
celula *EsvaziaLista(celula *lista){
    if (lista != NULL){
        celula *aux = lista->proxima; // aponta para a 1º Célula
        celula *noremov = NULL; // Célula que será removida
        while (aux != lista){
                noremov = aux; // Célula que será removida
                aux = aux->proxima;//Aponta para a próxima Célula
                delete noremov; // libera memória
        }
        delete aux; // libera memória (Última Célula)
        cout <<"\nLista esvaziada com sucesso!!!\n";
        return NULL;
    }
}
```

Testando a Lista...

```
int main(){
     // Esvazia a Lista
    lista = EsvaziaLista(lista);
    ImprimeLista(lista);
    return 0;
}
```

Inserir ordenado em uma Lista Circular



1. Aloca espaço para a nova Célula
2. Define os valores dos elementos da Célula
3. Percorre a Lista a procura da posição adequada para inserção...

Inserir ordenado em uma Lista Circular

```
celula *InsereOrdenado(celula *lista, int valor){
  if(lista == NULL) // A lista esta vazia
                  return inseriringicio(lista, valor);
          else // Existe(m) celulas(s) na Lista
          {
                  celula *aux = lista->proxima;
                  celula *ant = NULL;

                  while(aux != lista && aux->elemento < valor)
                  {
                          ant = aux;
                          aux = aux->proxima;
                  }
                  if(ant == NULL)
                  {
                          if(aux->elemento > valor)
                          {
                                  return inseriringicio(lista, valor);
                          }
                          else
                          {
                                  return inserirFim(lista, valor);
                          }
                  }
```

Inserir ordenado em uma Lista Circular

```
else if(aux == lista && aux->elemento < valor)
                {
                        celula *nova = new celula;
                        nova->elemento = valor;
                        nova->proxima = aux->proxima;
                        aux->proxima = nova;
                        return nova;
                }
                else
                {
                        celula *nova = new celula;
                        nova->elemento = valor;
                        nova->proxima = aux;
                        ant->proxima = nova;
                        return lista;
                }
        }
}
```

Testando a Lista...

```
int main(){
      // insere o elemento 29
   lista = InsereOrdenado(lista,n 29);
   ImprimeLista(lista);
   // insere o elemento 61
   lista = InsereOrdenado (lista, 61);
   ImprimeLista(lista);
   return 0;
}
```

Exercícios

- Desenvolva funções para:

(a) contar o número de elementos numa lista circular;

(b) ordenar uma lista circular

(c) concatenar duas listas circulares;

(d) intercalar duas listas ordenadas;

(e) fazer uma cópia da lista;