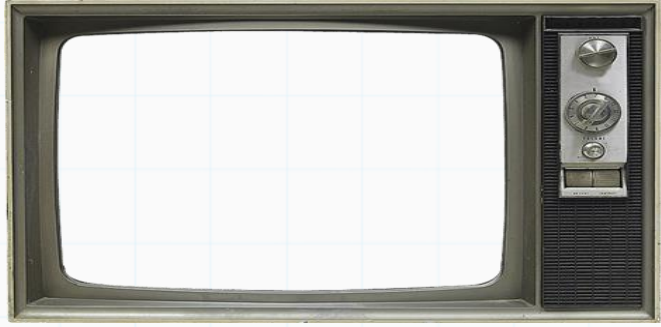
Programação Estruturada

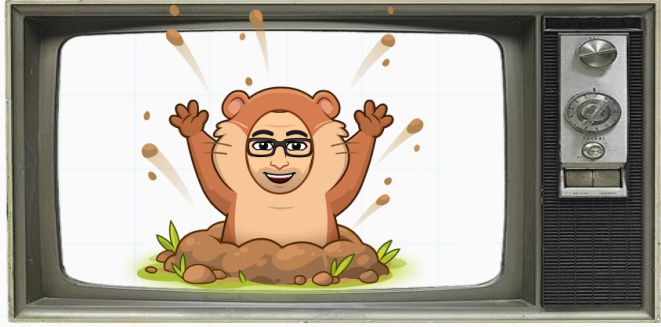


Professor : Yuri Frota

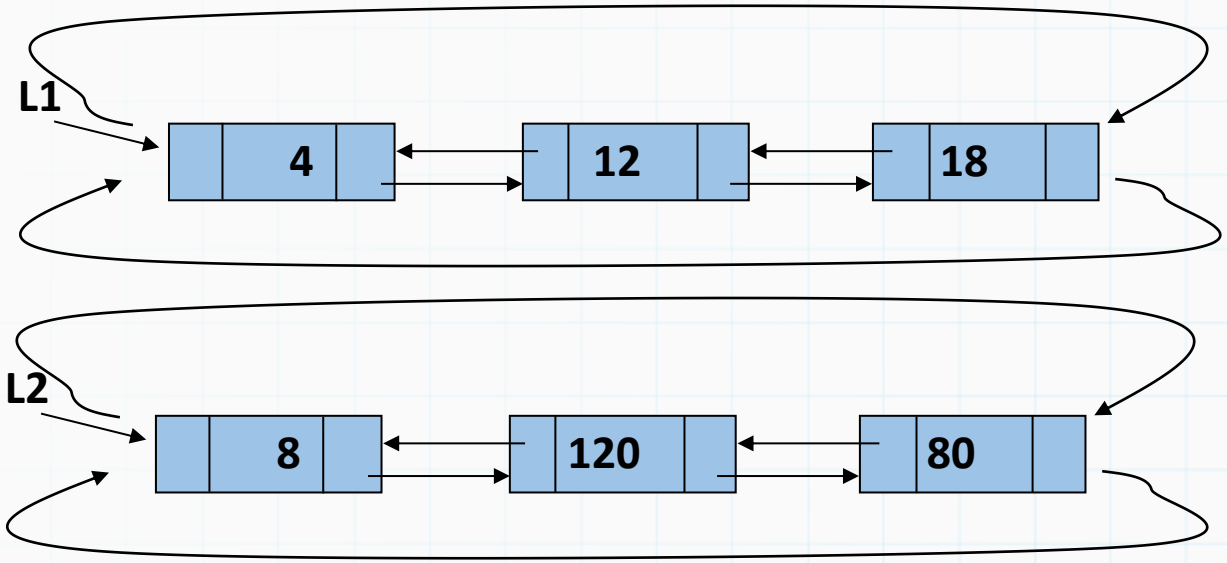
yuri@ic.uff.br

- Utilize os arquivos fornecidos main.c e tad.c com o TAD básico de listas circulares duplamente encadeadas para fazer as questões a seguir

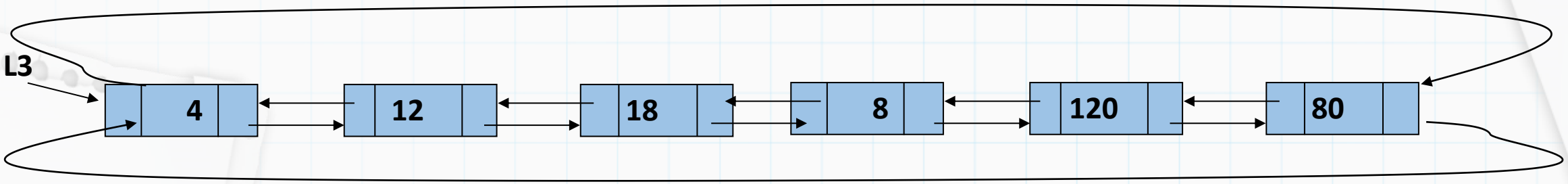
Listas circulares duplamente encadeadas - LAB



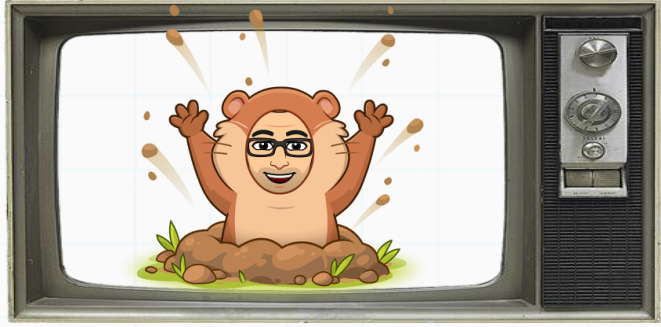
1) Dado duas listas circulares e duplamente encadeadas, escreva uma função para concatenar as duas listas em uma nova lista, apenas fazendo redirecionamento de ponteiros, sem utilizar/alocar estruturas auxiliares (vetores ou listas).



Veja exemplo da execução abaixo:



Listas circulares duplamente encadeadas - LAB



1) Dado duas listas circulares e duplamente encadeadas, escreva uma função para concatenar as duas listas em uma nova lista, apenas fazendo redirecionamento de ponteiros, sem utilizar/alocar estruturas auxiliares (vetores ou listas).

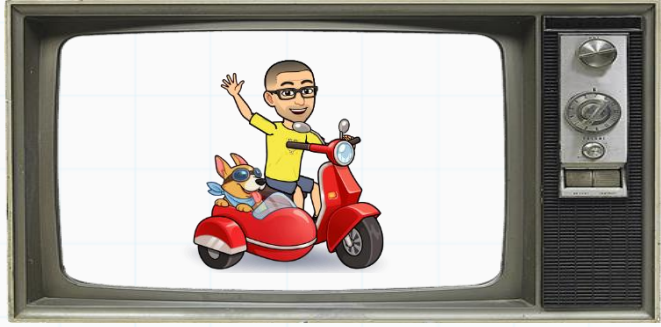
Código da main.c

```
int main()
{
    int k;

    lista *L1 = NULL;
    L1 = insere_lista(L1, 8);
    L1 = insere_lista(L1, 6);
    L1 = insere_lista(L1, 9);
    L1 = insere_lista(L1, 2);
    implime_lista(L1);

    lista *L2 = NULL;
    L2 = insere_lista(L2, 7);
    L2 = insere_lista(L2, 4);
    L2 = insere_lista(L2, 10);
    implime_lista(L2);

    lista *L3 = concatena_listas(L1,L2);
    implime_lista(L3);

    L3 = exclui_lista (L3);
    return 0;
}
```

Exemplo de execução:

```
L = 2, 9, 6, 8,
L = 10, 4, 7,
L = 2, 9, 6, 8, 10, 4, 7,
```

Listas circulares duplamente encadeadas - LAB



2) Dado uma lista circular e duplamente encadeada com n elementos, escreva uma função para remover um elemento na posição n-1>k>=0

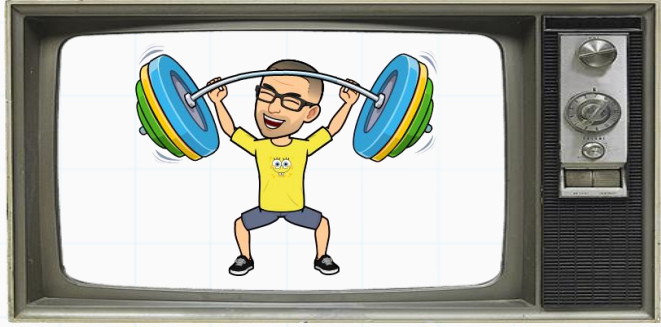
main.c

```
int main()
{
    lista *L1 = NULL;
    L1 = insere_lista_pos_DC(L1, 2, 0);
    L1 = insere_lista_pos_DC(L1, 6, 1);
    L1 = insere_lista_pos_DC(L1, 7, 2);
    L1 = insere_lista_pos_DC(L1, 23, 3);
    L1 = insere_lista_pos_DC(L1, 13, 4);
    imprime_lista_DC(L1);

    L1 = remove_lista_pos_DC(L1, 2);
    imprime_lista_DC(L1);
    L1 = remove_lista_pos_DC(L1, 4);
    imprime_lista_DC(L1);
    L1 = remove_lista_pos_DC(L1, 3);
    imprime_lista_DC(L1);
    L1 = remove_lista_pos_DC(L1, 0);
    imprime_lista_DC(L1);
    L1 = remove_lista_pos_DC(L1, 0);
    imprime_lista_DC(L1);
    L1 = remove_lista_pos_DC(L1, 0);
    imprime_lista_DC(L1);

    exclui_lista_DC(L1);
    return 0;
}
```

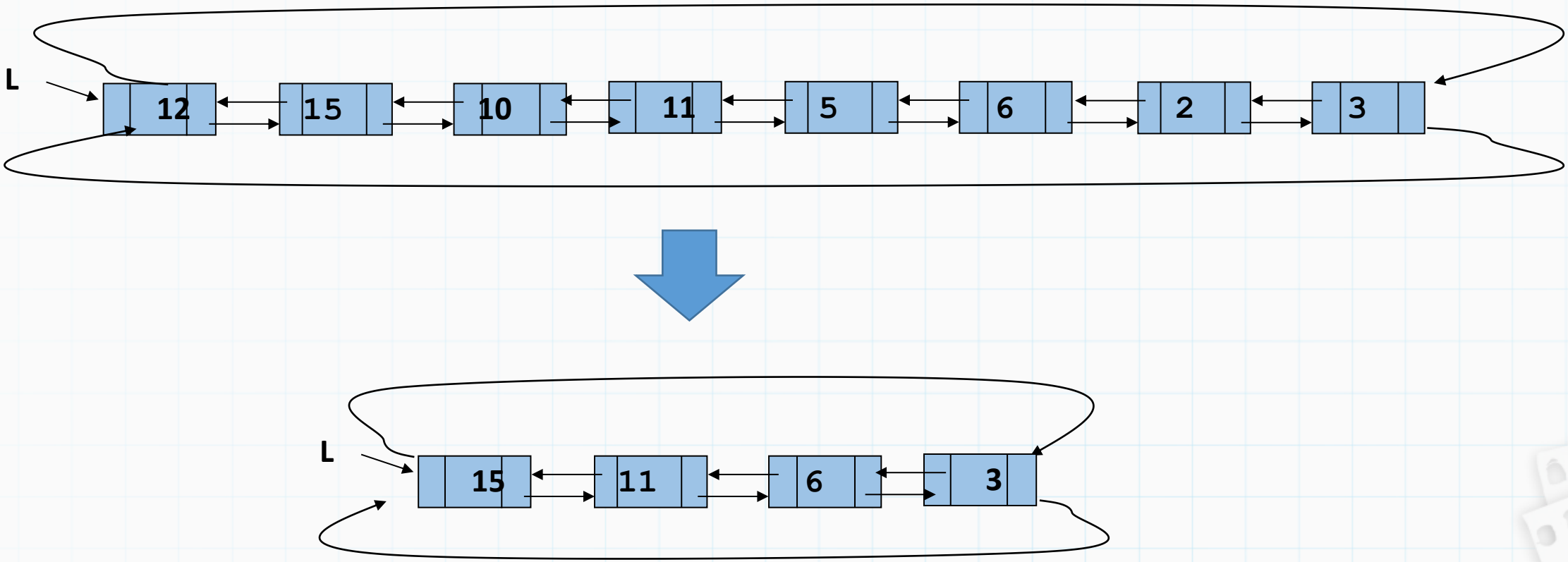
Execução

```
L = 2, 6, 7, 23, 13,
L = 2, 6, 23, 13,
Posicao invalida
L = 2, 6, 23, 13,
L = 2, 6, 23,
L = 6, 23,
L = 23,
L = vazio
```

Listas circulares duplamente encadeadas - LAB

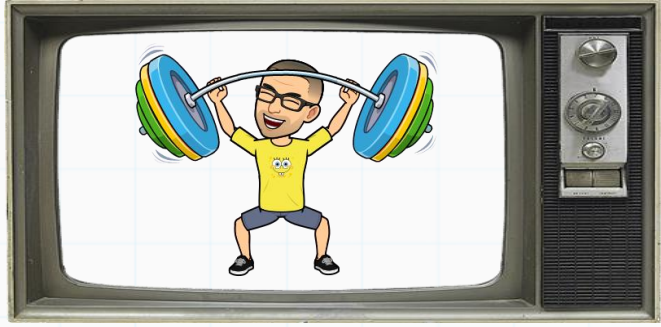


3) Dado uma lista circular e duplamente encadeada com n elementos, escreva uma função para remover os elementos da lista em que o próximo elemento a ele é maior que ele, sem utilizar/alocar estruturas auxiliares (vetores ou listas).



Veja exemplo de execução a seguir

Listas circulares duplamente encadeadas - LAB



3) Dado uma lista circular e duplamente encadeada com n elementos, escreva uma função para remover os elementos da lista em que o próximo elemento a ele é maior que ele, sem utilizar/alocar estruturas auxiliares (vetores ou listas).

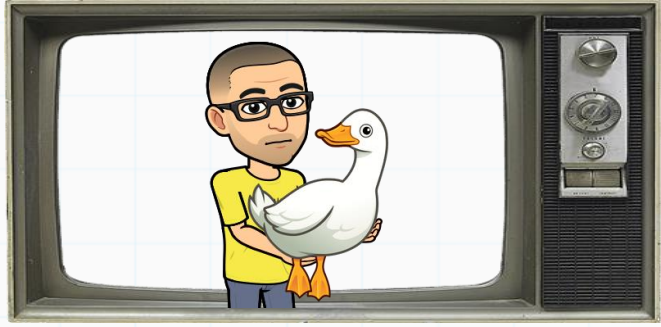
main.c

```
int main()
{
    lista *L1 = NULL;
    L1 = insere_lista_pos_DC(L1, 12, 0);
    L1 = insere_lista_pos_DC(L1, 15, 1);
    L1 = insere_lista_pos_DC(L1, 10, 2);
    L1 = insere_lista_pos_DC(L1, 11, 3);
    L1 = insere_lista_pos_DC(L1, 5, 4);
    L1 = insere_lista_pos_DC(L1, 6, 5);
    L1 = insere_lista_pos_DC(L1, 2, 6);
    L1 = insere_lista_pos_DC(L1, 3, 7);
    imprime_lista_DC(L1);

    L1 = remove_proximos_maiores(L1);
    imprime_lista_DC(L1);

    exclui_lista_DC(L1);
    return 0;
}
```

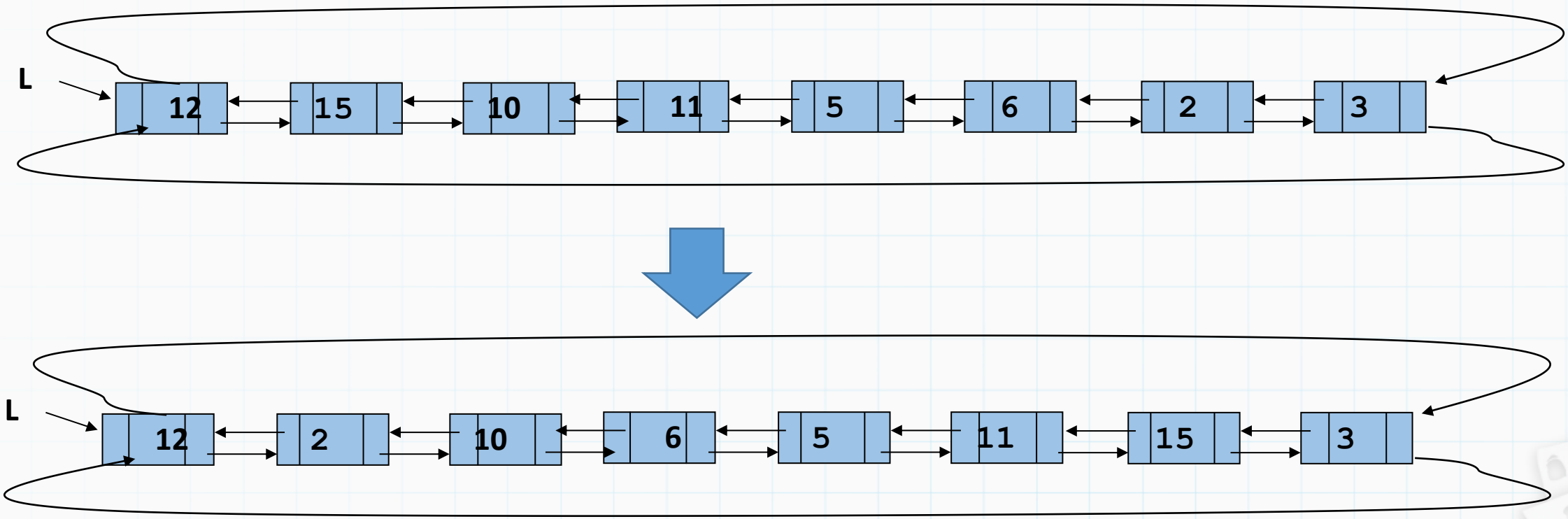
Execução

```
L = 12, 15, 10, 11, 5, 6, 2, 3,
remove onde proximos maiores
   12 menor que 15, remove
   10 menor que 11, remove
   5 menor que 6, remove
   2 menor que 3, remove
L = 15, 11, 6, 3,
```

Listas circulares duplamente encadeadas - LAB

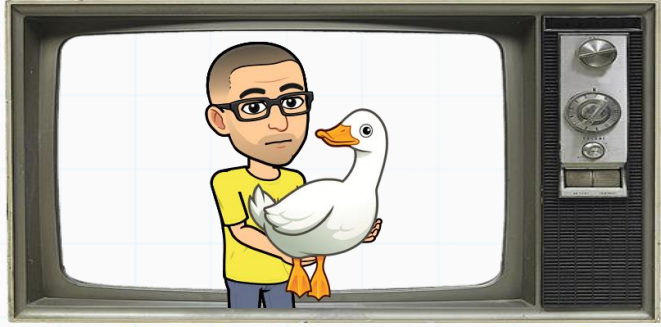


4) Dado uma lista circular e duplamente encadeada escreva uma função para colocar todos os elementos pares antes dos impares, sem utilizar/alocar estruturas auxiliares (vetores ou listas).



Veja exemplo de execução a seguir

Listas circulares duplamente encadeadas - LAB



4) Dado uma lista circular e duplamente encadeada escreva uma função para colocar todos os elementos pares antes dos impares, sem utilizar/alocar estruturas auxiliares (vetores ou listas).

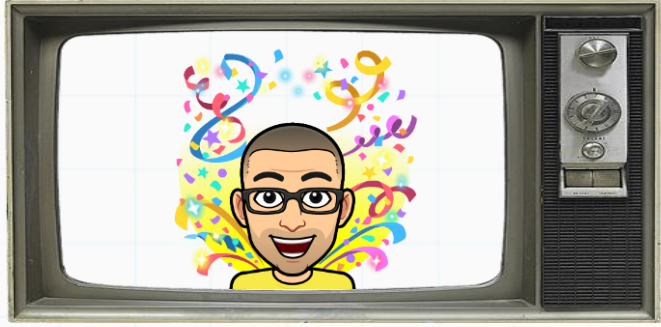
main.c

```
int main()
{
    lista *L1 = NULL;
    L1 = insere_lista_pos_DC(L1, 17, 0);
    L1 = insere_lista_pos_DC(L1, 15, 1);
    L1 = insere_lista_pos_DC(L1, 8, 2);
    L1 = insere_lista_pos_DC(L1, 12, 3);
    L1 = insere_lista_pos_DC(L1, 10, 4);
    L1 = insere_lista_pos_DC(L1, 5, 5);
    L1 = insere_lista_pos_DC(L1, 4, 6);
    L1 = insere_lista_pos_DC(L1, 1, 7);
    L1 = insere_lista_pos_DC(L1, 7, 8);
    L1 = insere_lista_pos_DC(L1, 6, 9);
    imprime_lista_DC(L1);

    printf("par impar\n");
    L1 = par_impar_DC(L1);
    imprime_lista_DC(L1);

    exclui_lista_DC(L1);
    return 0;
}
```

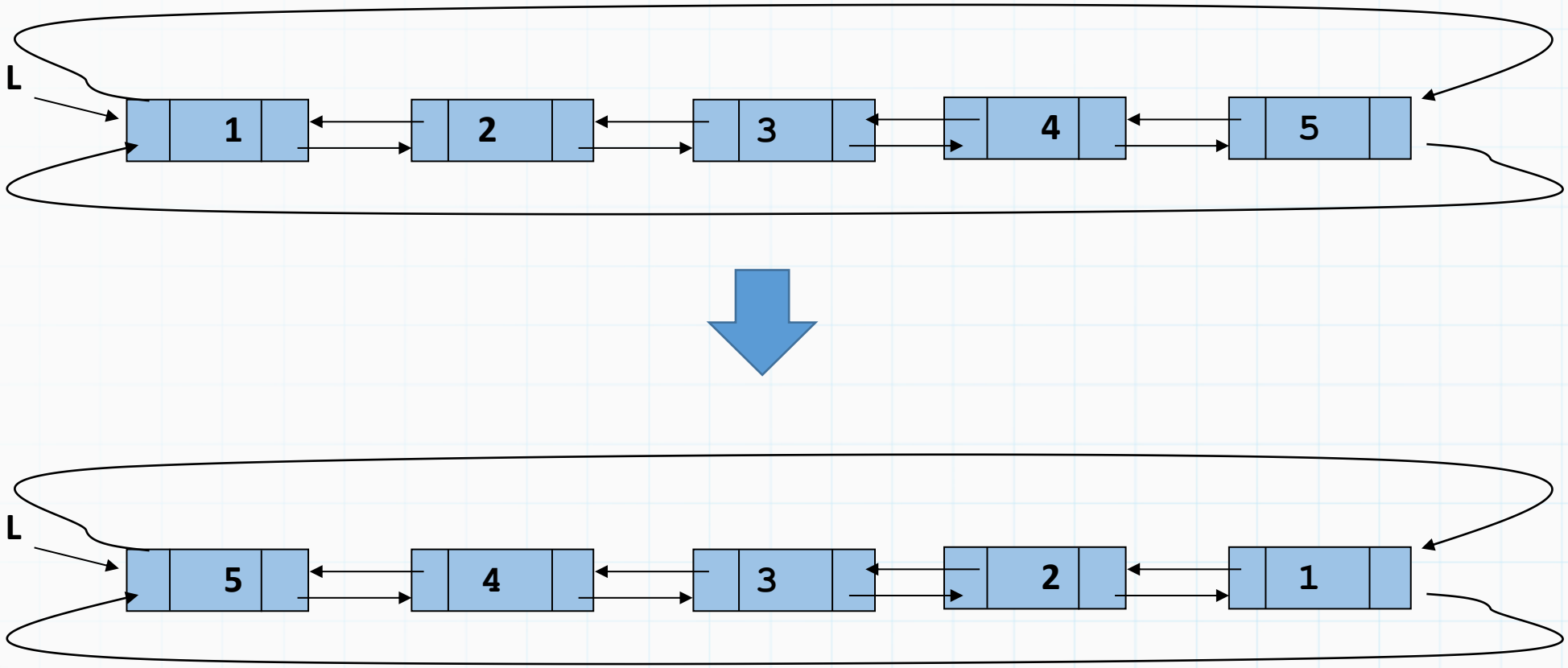
Execução

```
L = 17, 15, 8, 12, 10, 5, 4, 1, 7, 6,
par impar
L = 6, 4, 8, 12, 10, 5, 15, 1, 7, 17,
```

Listas circulares duplamente encadeadas - LAB

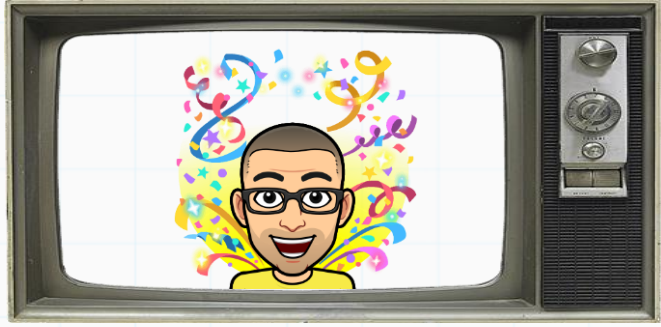


5) Dado uma lista circular e duplamente encadeada escreva uma função para inverter os elementos da lista, apenas fazendo reatribuição de ponteiros.



Exemplo de execução a seguir:

Listas circulares duplamente encadeadas - LAB



5) Dado uma lista circular e duplamente encadeada escreva uma função para inverter os elementos da lista, apenas fazendo reatribuição de ponteiros.

main.c

```
int main()
{
    lista *L1 = NULL;
    L1 = insere_lista_pos_DC(L1, 1, 0);
    L1 = insere_lista_pos_DC(L1, 2, 1);
    L1 = insere_lista_pos_DC(L1, 3, 2);
    L1 = insere_lista_pos_DC(L1, 4, 3);
    L1 = insere_lista_pos_DC(L1, 5, 4);
    imprime_lista_DC(L1);

    printf("inverte\n");
    L1 = inverte_lista_DC(L1);
    imprime_lista_DC(L1);

    exclui_lista_DC(L1);
    return 0;
}
```

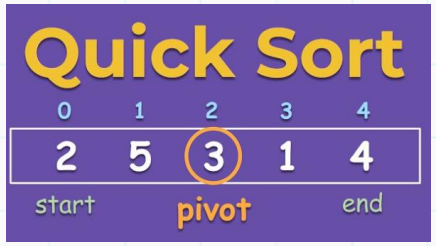
Execução

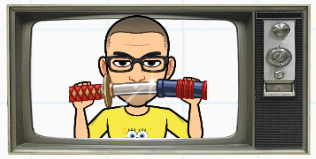
```
L = 1, 2, 3, 4, 5,
inverte
L = 5, 4, 3, 2, 1,
```

Listas circulares duplamente encadeadas - LAB

```c
void quickSort(int* v, int ini, int fim)
{
    if (ini < fim)
    {
        int pivo_ind = particao(v, ini, fim);

        quickSort(v, ini, pivo_ind - 1);
        quickSort(v, pivo_ind + 1, fim);
    }
}
```

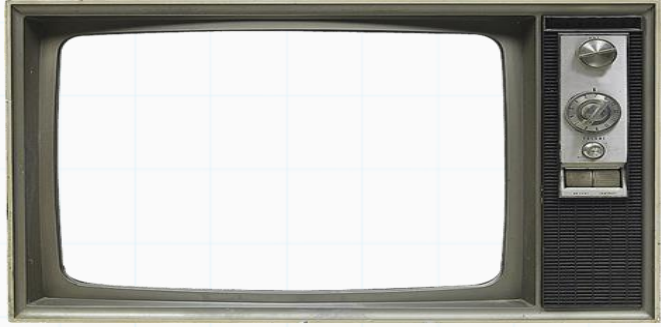
6) Estudamos anteriormente o método de ordenação do Quicksort em vetores, vamos agora implementar uma função que dado uma lista circular duplamente encadeada, ordene a lista pelo método do Quicksort, sem utilizar/alocar estruturas auxiliares (vetores ou listas).





```c
int particao(int* v, int ini, int fim)
{
    int pivo = v[ini];
    int i    = ini;
    int j    = fim;

    while (i < j)
    {
        while (v[i] <= pivo && i <= fim - 1)
            i++;

        while (v[j] > pivo && j >= ini + 1)
            j--;

        if (i < j)
        {
            int temp = v[i];
            v[i]     = v[j];
            v[j]     = temp;
        }
    }
    int temp = v[ini];
    v[ini]   = v[j];
    v[j]     = temp;
    return j;
}
```

Até a próxima



Slides baseados no curso de Aline Nascimento