Universidade Estadual de Santa Cruz
Disciplina: Estrutura de Dados
Professora: Jacqueline Midlej
Data: 26/10/2023
Nome: _____
Nº matrícula: _____

Prova 1

1. **(1.5 pontos)** Sabe-se que em uma estrutura de dados do tipo fila, a remoção ocorre pela extremidade da esquerda e a inserção pela extremidade da direita. Dada uma fila simples (não-circular), pode ocorrer a seguinte configuração que indica "fila cheia" ao tentar inserir o item I.

0	1	2	3	4	5	6	7
					F	G	H

front=0; rear=7;

Uma possível solução é: sempre que tentar inserir um elemento e rear atingiu o último elemento no vetor, iremos deslocar toda a fila para trás, iniciando em 0, e depois fazer a inserção. Como na imagem:

0	1	2	3	4	5	6	7
F	G	H	I				

front=0; rear=3;

Faça a rotina de inserção na fila contendo essa solução. Considere uma fila apenas com 8 elementos. Use a seguinte estrutura em C **ou** a seguinte Classe em Python para a sua implementação:

```
struct fila {
      int rear, front;
      char items[8];
};
```

```
class fila:
    def __init__(self):
        self.front = 0
        self.rear = -1
        self.items = [None]*7
```

```
void insert (int x , fila aux){
        // descolamento
        if (aux.rear==7 && aux.front>0)
                for (i=aux.front, j=0; i<=aux.rear; i++, j++)
                        aux.items[j]=a.items[i];
        // inserção normal
        a.items[++a.rear]=x;
}
```

2. **(1.0 ponto)** O que faz a função a seguir? OBS: mesma funcionalidade em diferentes linguagens.

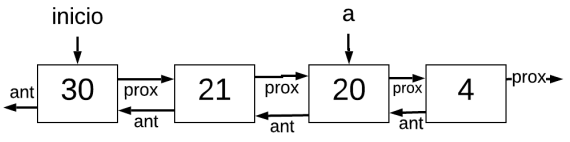
resposta: verifica se lista está ordenada em ordem decrescente

```
///////////////// EM C
struct no {
      int valor;
      struct no * prox;
};

typedef struct no* pno;

int func (pno inicio){
    pno atual=inicio;
    if (atual != NULL) {
        while (atual->prox!=NULL){
            if (atual->prox->valor > atual->valor)
                return 0;
            atual=atual->prox;
        }
    }
    return 1;
}


######### EM PYTHON
class no:
    def __init__(self, valor, prox):
        self.valor=valor
        self.prox=None

def func (inicio: no)->int:
    atual=inicio
    if atual is not None:
        while atual.prox is not None:
            if atual.prox.valor > atual.valor:
                return 0
            atual=atual.prox
    return 1
```

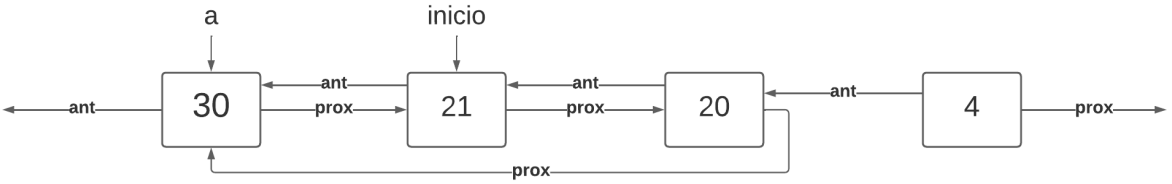
3. **(1.0 ponto)** Para cada alternativa abaixo, execute as instruções de código na lista e mostre o desenho final da lista. **Indique/desenhe todas as variáveis.**

a)

```
a.prox=inicio;
while (inicio.prox!=a)
     inicio=inicio.prox;
while (a.ant !=NULL)
     a=a.ant;
```

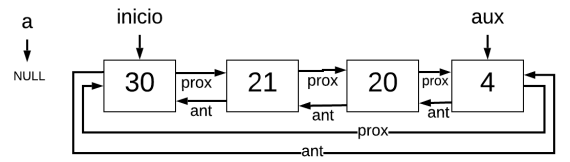


Resposta:

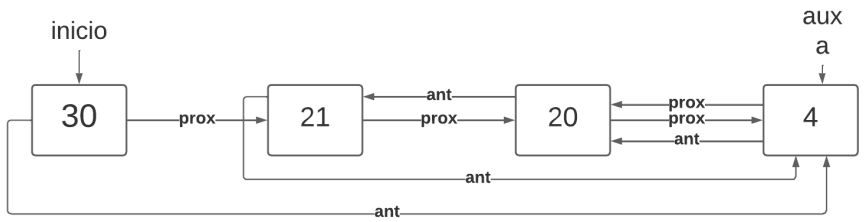


b)

```
inicio.ant=aux.ant.prox;
aux.prox.prox.ant=aux;
a=inicio.prox.ant;
aux.prox=inicio.prox.ant.ant;
```



Resposta:



4. **(3.0 pontos)** Escreva uma função para **deslocar o nó da posição i para frente** em uma lista. Por exemplo, dada a lista 10->20->21->22->, desloca-se o nó da posição 1 para frente, resultando em 10->21->20->22->. Escolha qualquer lista **encadeada** de sua preferência, simples ou duplamente encadeada, circular ou não. Considere que o nó a ser deslocado é um nó do meio (existe nós antes e depois dele) e considere que a posição i sempre existe na lista.

Obs: **deslocar o nó significa manipular os ponteiros**, não fazer uma cópia do valor inteiro, ou adicionar novos nós à lista. Pode ser implementada em C ou Python. Defina a classe ou estrutura que você irá utilizar para implementar a função.

5. **(1.5 ponto)** Escreva uma função para **comparar tamanhos de 2 listas encadeadas**. A função deve receber o início de duas listas e retornar 0 se elas têm o mesmo tamanho, -1 se a primeira é maior que a segunda, 1 se a segunda é maior que a primeira. Você pode implementar em C ou Python e escolher qualquer tipo de **lista encadeada**. Mostre a estrutura ou classe que usou para implementar sua função, isto é, quais as variáveis do seu nó.

uma possível solução:

```
class no:
      def __init__(self, valor, prox):
            self.valor=valor
            self.prox=prox

def comparar (inicio_1: no, inicio_2: no):
      while inicio_1 is not None and inicio_2 is not None:
            inicio_1=inicio_1.prox
            inicio_2=inicio_2.prox
      if inicio_1 is None and inicio_2 is None:
            return 0
      elif inicio_1 is not None:
            return -1
      else
            return 1
```

6. **(2.0 ponto)** A seguinte função **concatena duas listas duplamente encadeadas circular**, onde a primeira lista conterá a concatenação das duas. Complete as lacunas para que a função realize o procedimento corretamente. Escolha uma das duas linguagens C ou Python. Considere que as duas listas têm pelo menos 1 elemento em cada. **Nenhuma delas é vazia.**

```
// Implementação em C
struct no {
      int valor;
      struct no * prox;
      struct no * ant;
};
typedef struct no * pno;

void concatCirc (pno *inicio1, pno *inicio2) {
      pno ultimo1=*inicio1;
      pno ultimo2=*inicio2;

      while (ultimo1->prox!=*inicio1)
            ultimo1=ultimo1->prox;
      ultimo1->prox=*inicio2;
      (*inicio2)->ant=ultimo1;

      while (ultimo2->prox!=*inicio2)
            ultimo2=ultimo2->prox;
      ultimo2.prox=*inicio1;
      (*inicio1)->ant=ultimo2;
}
```

```
# Implementação em Python
class no:
      def __init__(self, valor , ant, prox):
            self.valor=valor
            self.ant=ant
            self.prox=prox

def concatCirc (inicio1: no, inicio2: no):
      ultimo1=inicio1
      ultimo2=inicio2

      while (ultimo1.prox!=inicio1):
            ultimo1=ultimo1.prox
      ultimo1.prox=inicio2;
      inicio2.ant=ultimo1;

      while (ultimo2.prox!=inicio2):
            ultimo2=ultimo2.prox

      ultimo2.prox=inicio1;
      inicio1.ant=ultimo2;
```

Boa prova!!