1) Dada as classes abaixo, explique qual estrutura de dados elas pertencem, o funcionamento de cada método e realize pelo menos 5 simulações para cada um deles.

```
public class NO {
    public int dado;
    public NO prox;
    public NO anterior;

    public NO(int e){
        dado=e;
        prox=null;
        anterior=null;
    }
}
```

```
public class Estrutura {
    private NO inicio;

    public Estrutura() {
        inicio=null;
    }

    public void AdicionaInicio(int e){
        NO n=new NO(e);
        if (inicio!=null){
            n.prox=inicio;
            inicio.anterior=n;
        }
        inicio=n;
    }


  public void AdicionaFinal(int e){
      NO n=new NO(e);
      if (inicio==null){
          inicio=n;
          n.anterior=null;
          n.prox=null;
      }
      else{
          NO aux=inicio;
          while(aux.prox!=null){
              aux=aux.prox;
          }
          aux.prox=n;
          n.anterior=aux;
          n.prox=null;
      }
  }
```

```
    public int RemoveInicio(){
        int r=-1;
        if (inicio==null){
            System.out.println("Lista vázia");
        }
        else{
            r=inicio.dado;
            inicio=inicio.prox;
            if (inicio!=null){
                inicio.anterior=null;
            }
        }
        return r;
    }


    public int RemoveFinal(){
        int r=-1;
        if (inicio==null){
            System.out.println("Lista vázia");
        }
        else
            if (inicio.prox==null){
                r=inicio.dado;
                inicio=null;
            }
            else{
                NO aux1=inicio;
                NO aux2=inicio;
                while(aux1.prox!=null){
                    aux2=aux1;
                    aux1=aux1.prox;
                }
                r=aux1.dado;
                aux1.anterior=null;
                aux2.prox=null;
            }
        return r;
    }



     public String percorre(){
         String r=" ";
         NO aux=inicio;
         while(aux!=null){
             r=r+"\n"+aux.dado;
             aux=aux.prox;
         }
         return r;
     }
}
```

2) Dada as classes abaixo, explique qual estrutura de dados elas pertencem, o funcionamento de cada método e explique a principal diferença entre os métodos AdicionaFinal e RemoveFinal em relação aos métodos apresentados no exercício 1.

```
public class NO {
    public int dado;
    public NO prox;
    public NO anterior;

    public NO(int e){
        dado=e;
        prox=null;
        anterior=null;
    }
}


public class Estrutura {

    private NO inicio;

    public Estrutura(){
        inicio=null;
    }

    public boolean Vazia(){
        return inicio==null;
    }

    public void AdicionaInicio(int e){
        NO n=new NO(e);

        if(Vazia()==false){
            n.prox=inicio;
            inicio.anterior=n;
        }
        inicio=n;
    }


  public void AdicionaFinal(int e){
      NO n=new NO(e);

      if(Vazia()==true){
          inicio=n;
          n.prox=null;
          n.anterior=null;
      }
      else{
          NO aux=BuscaUltimo(inicio);
          aux.prox=n;
          n.anterior=aux;
          n.prox=null;
      }
  }
```

```
public NO BuscaUltimo(NO aux){
    if(aux.prox!=null){
        return BuscaUltimo(aux.prox);
    }
    return aux;
}

public int RemoveInicio(){
    int r=-1;

    if(Vazia()==true){
        JOptionPane.showMessageDialog(null,"Lista Vázia");
    }
    else{
        r=inicio.dado;
        inicio=inicio.prox;

        if(inicio!=null){
            inicio.anterior=null;
        }
    }
    return r;
}

public int RemoveFinal(){
    int r=-1;

    if(Vazia()==true){
        JOptionPane.showMessageDialog(null,"Lista Vázia");
    }

    else{
        if(inicio.prox==null){
            r=inicio.dado;
            inicio=null;
        }
        else{
            NO aux2=LocalizaDadoParaRemocao(inicio,inicio);
            r=aux2.prox.dado;
            aux2.prox=null;
        }
    }
    return r;
}
```

```
    public NO LocalizaDadoParaRemocao(NO aux1, NO aux2){
        if(aux1.prox!=null){
            return LocalizaDadoParaRemocao(aux1.prox,aux1);
        }
        return aux2;
    }


    public void percorre(){
        NO aux=inicio;
        String r=" ";
        JOptionPane.showMessageDialog(null,"Lista:"+ConcatenaValores(aux,r));
    }

    public String ConcatenaValores(NO aux, String r){
        if(aux!=null){
            r=r+"\n"+aux.dado;
            return ConcatenaValores(aux.prox,r);
        }
        return r;
    }
}
```

3) Explique cada uma das afirmações abaixo, descrevendo em qual estrutura de dados elas pertencem. Justifique sua resposta.

I - Inserção e remoção de elementos acontecem apenas na “cabeça” da estrutura.

II - Inserção de um nó no meio da estrutura pode ser realizada com custo computacional constante.

III - Respeito à política FIFO: o primeiro elemento que entra é o primeiro a sair.

4) Explique se a frase abaixo se refere a uma pilha, lista encadeada, fila, matriz ou vetor. Justifique sua resposta.

“Na alocação dinâmica de memória, os dados são armazenados em posições de memória referenciadas e dispostos em uma dada organização não linear, sendo possível, a partir de um elemento, encontrar os próximos.”

5) Considerando as definições para listas, pilhas e filas, explique cada uma das frases abaixo. Justifique se cada uma das afirmações é verdadeira ou falsa.

A) Uma lista é um tipo de fila que se caracteriza por considerar que o primeiro elemento a entrar é o primeiro a sair.

B) Lista é um conjunto de filas e pilhas e se compõe por elementos que podem ser ligados ou não.

C) Lista é uma sequência finita de elementos ligados entre si. Podem ser organizada de tal forma que implemente uma fila ou uma pilha.

6) Implemente uma Lista de Alunos em alocação dinâmica de memória duplamente encadeada com os atributos id, nome e curso, **que utilize somente métodos recursivos em todas as implementações que envolvam estruturas de repetições.** Realize as seguintes operações

- verificar se a lista está vazia, retornando true se estiver vazia e false se não estiver;
- adicionar um aluno no início da lista, caso a operação não possa ser realizada, mostre mensagem avisando;
- adicionar um aluno no final da lista, caso a operação não possa ser realizada, mostre mensagem avisando;
- remover um aluno do início da lista, retornando o elemento que foi removido, caso a operação não possa ser realizada, mostre mensagem avisando.
- remover um aluno do final da lista, retornando o elemento que foi removido, caso a operação não possa ser realizada, mostre mensagem avisando.
- ordenar a lista por meio do bubble sort pelo nome do aluno para a lista original;
- implementar um método de busca sequêncial para retornar pelo nome do aluno
- percorrer e apresentar cada um dos elementos da lista