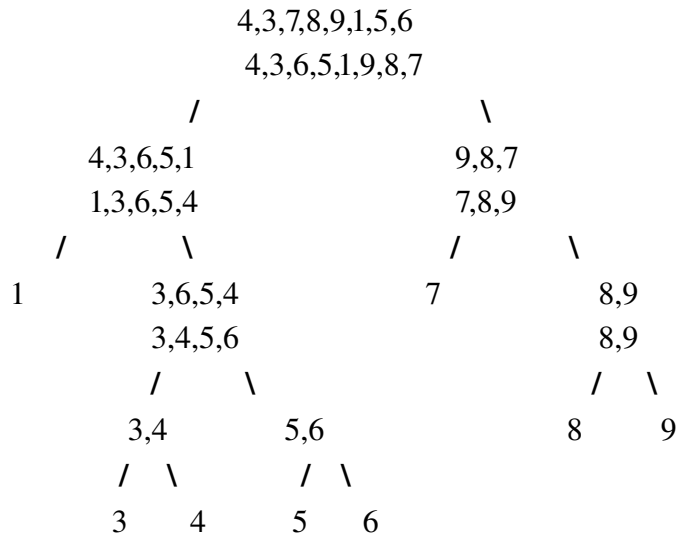
Prova de Introdução a Ciência da Computação II
Prof. Eduardo C. Xavier

Questão 1 (10/3 pontos) – Faça a execução do algoritmo QuickSort para o vetor (4,3,7,8,9,1,5,6) mostrando a árvore de recursão, e indique a ordem em que as chamadas recursivas e retornos são executados. Considere como pivô sempre o último elemento.



Questão 2 (10/3 pontos) – Faça a implementação da interface IFila abaixo usando nós da classe dada:

```
public interface Ifila<T>{
  //insere objeto no fim da fila
  public void insere(T o);
  //remove objeto do inicio da fila e o retorna
  public T remove();
}
```

```
class No<T>{
    No<T> ant=null;
    No<T> prox=null;
    T elemento=null;
}
```

```
public class Fila<T> implements Ifila<T>{
    No<T> inicio, fim;

    Fila<T>(){ //construtor
        inicio = new No<T>();
        fim = new No<T>();
        inicio.prox = fim;
        fim.ant = inicio;
    }

    public void insere(T o){
        No<T> novo = new No<T>();
        novo.elemento = o;
```

```
            novo.ant = fim.ant;    //insere nó no fim da fila
            novo.prox = fim;
            fim.ant.prox = novo;
            fim.ant = novo;
        }

        public T remove(){
             if(inicio.prox!=fim){ //se tiver algum elemento
                  T aux = inicio.prox.elemento;
                  inicio.prox.prox.ant = inicio;
                  inicio.prox = inicio.prox.prox;
                  return aux;
             }
             else return null;
        }

}
```

Questão 3 (10/3 pontos) – a) Considere o seguinte problema: Como entrada temos um vetor de inteiros e um valor inteiro x. O problema consiste em determinar se há dois elementos do vetor cuja soma seja x. Explique em no máximo 5 linhas como responder ao problema em tempo O(n log n) no pior caso, onde n é o tamanho do vetor.

Ordene o vetor com o MergeSort (O(n log n)) e faça um laço onde para cada elemento v[i] do vetor procuramos por x-v[i]. A busca é feita utilizando-se a busca binária.

b) Escreva um método em Java que resolve o problema do item (a) e que tenha tempo de execução no pior caso de O(n log n). Faça a análise do seu algoritmo e explique o porquê da complexidade. Dica: Você pode assumir a existência dos algoritmos de ordenação e busca vistos em aula.

```
public boolean acha_resposta(int[] v, int x){
   MergeSort(v);  //ordena para podermos fazer busca binária
  for(int i=0; i<v.length;i++)
       if(buscaBinaria(v,x-v[i]) )
            return true;
  return false;
}
```