MAUÁ	<CdDisc> - <PerDisc> <Exer> - <Sem>o. semestre - <NomeIdProva>	<NoSeq> <IdSala>
Disc.: <NOMEDISC>	Cód. Disciplina: CIC203	
Curso: <NomeCurDisc>		Série: <Se

Aluno: <Aluno>				
Curso: <Cur>	Série <SerCur>ª.	Período: <PerCur>	RG: <RG>	
São Caetano do Sul, 07 de junho de 2024.			RA: <RA>	
Assinatura aluno:			Nota:	

INSTRUÇÕES GERAIS

- Duração da prova: 90 minutos.
- Tempo mínimo de permanência na sala de prova: 30 minutos.
- Não é permitido ausentar-se da sala para posterior retorno.

ORIENTAÇÕES ESPECÍFICAS DESTA PROVA

1. **Leia atentamente <u>todas</u> as instruções antes do início da prova e depois cada enunciado de questão.**
2. Não é permitido o uso de celulares, *smart watch*es, pontos eletrônicos e estojos.
3. Todo material de consulta está na própria prova.
4. As questões podem ser resolvidas a lápis. Se fizer rascunhos, só identifique com a palavra **rascunho**, não é necessário apagar. Desenhos também podem ser deixados.
5. O valor de cada questão é 2,5 pontos.

Rascunhos

1. Veja uma implementação de fila sobre vetores, realizada em aula. Nela o vetor é uma estrutura circular, para que seja possível aproveitar melhor a alocação em memória, sem a necessidade de realizar deslocamentos dos valores, o que aumentaria o custo computacional das operações de inserção e remoção.

```
public class Fila {
  private int[] dados;
  private int primeiro;
  private int ultimo;
  private int tamanho;
  public Fila (int capacidade) {
    if (capacidade > 5)
      dados = new int[capacidade];
    else
      dados = new int[5];
    primeiro = 0;
    ultimo = dados.length - 1;
    tamanho = 0;
  }
  public Fila () {
    this(5);
  }
  public int getTamanho() {
    return tamanho;
  }
  public boolean estaVazia () {
    return tamanho == 0;
  }
  public boolean estaCheia () {
    return tamanho == dados.length;
  }
  int proximaPosicao (int posicao) {
    return (posicao + 1) % dados.length;
  }
```

```
  public boolean enfileira (int i) {
    if (estaCheia()) return false;
    ultimo = proximaPosicao(ultimo);
    dados[ultimo] = i;
    tamanho++;
    return true;
  }

  public int desenfileira () {
    if (estaVazia()) return -1;
    int temp = dados[primeiro];
    primeiro = proximaPosicao(primeiro);
    tamanho--;
    return temp;
  }

  @Override
  public String toString () {
    if (estaVazia()) return "fila vazia";
    String s = "";
    int cont = primeiro;
    do {
      s = s + dados[cont] + " ";
      cont = proximaPosicao(cont);
    } while (cont != proximaPosicao(ultimo));
    return s;
  }
```

Agora, considere as operações a seguir e “tire uma foto” (na folha seguinte, estão mapeadas as operações) da estrutura em memória, para cada operação:

```
public static void main (String[] args) {
  01  Fila f = new Fila();
  02  f.enfileira (3)
  03  f.enfileira (8);
  04  f.enfileira (5);
  05  f.desenfileira ();
  06  f.enfileira (4);
  07  f.enfileira (6);
  08  f.desenfileira ();
  09  f.desenfileira ();
  10  f.enfileira (7)
  11  f.enfileira (6);
  12  f.enfileira (1);
  13  f.desenfileira ();
  14  f.desenfileira ();
}
```

Você pode deixar os espaços do vetor que “não têm dados” em branco.

01	primeiro		dados							
	ultimo			0	1	2	3	4		
	tamanho									
02	primeiro		dados							
	ultimo			0	1	2	3	4		
	tamanho									
03	primeiro		dados							
	ultimo			0	1	2	3	4		
	tamanho									
04	primeiro		dados							
	ultimo			0	1	2	3	4		
	tamanho									
05	primeiro		dados							
	ultimo			0	1	2	3	4		
	tamanho									
06	primeiro		dados							
	ultimo			0	1	2	3	4		
	tamanho									
07	primeiro		dados							
	ultimo			0	1	2	3	4		
	tamanho									
08	primeiro		dados							
	ultimo			0	1	2	3	4		
	tamanho									
09	primeiro		dados							
	ultimo			0	1	2	3	4		
	tamanho									
10	primeiro		dados							
	ultimo			0	1	2	3	4		
	tamanho									
11	primeiro		dados							
	ultimo			0	1	2	3	4		
	tamanho									
12	primeiro		dados							
	ultimo			0	1	2	3	4		
	tamanho									
13	primeiro		dados							
	ultimo			0	1	2	3	4		
	tamanho									
14	primeiro		dados							
	ultimo			0	1	2	3	4		
	tamanho									

2. Considere a implementação de fila (de inteiros) sobre lista ligada dada a seguir.

```
public class FilaDinamica {
    private class No {
        private int info;
        private No prox;
        public No (int i){
            info = i;
            prox = null;
        }
        public int getInfo() {
            return info;
        }
        public No getProx() {
            return prox;
        }
        public void setInfo (int info) {
            this.info = info;
        }
        public void setProx (No prox) {
            this.prox = prox;
        }
        @Override
        public String toString () {
            return "|" + info + "|->";
        }
    }
    private No prim;
    private No ult;
    private int tamanho;
    public boolean estaVazia() {
        return tamanho == 0;
    }
    public void enfileira (int i) {
        No novo = new No(i);
        if (estaVazia())
            prim = novo;
        else
            ult.setProx(novo);
        ult = novo;
        tamanho++;
    }
    public int desenfileira () {
        int i = prim.getInfo();
        tamanho--;
        prim = prim.getProx();
        if (prim == null) //esvaziou a fila
            ult = null;
        return i;
    }
    @Override
    public String toString () {
        String s = "fila: ";
        if (estaVazia()) s += "vazia";
        else {
            No runner = prim;
            while (runner != null) {
                s += runner + " ";
                runner = runner.getProx();
            }
            s += "//";
        }
        return s;
    }
}
```

Inspirando-se no projeto do estacionamento, desenvolver um método na classe fila que encontra um valor x (passado como parâmetro) e o coloca no final da fila. Tente fazê-lo o mais eficiente possível.

Resposta

Rascunhos

3. Sabemos que o nível de um nó em uma árvore binária de busca é dado por: a raiz da árvore é dita estar no nível zero e, se um nó estiver no nível k, seus filhos estão no nível k+1. Considere as partes de interesse da implementação básica de uma árvore binária de busca, realizada em aula, dadas a seguir.

```
public class ABB {
  private No raiz;
  public boolean estaVazia () {
    return raiz == null;
  }
  public int nivel () {
    if (estaVazia()) return 0;
    return nivelRec(raiz);
  }
  int nivelRec (No atual) {
    if (atual.getEsquerda() == null && atual.getDireita() == null) return 0;
    int nivelEsq=0, nivelDir=0;
    if (atual.getEsquerda() != null) nivelEsq = nivelRec(atual.getEsquerda());
    if (atual.getDireita() != null) nivelDir = nivelRec(atual.getDireita());
    if (nivelEsq > nivelDir) return nivelEsq + 1;
    else return nivelDir + 1;
  }
  public boolean buscaBinaria (int x) {
    if (estaVazia()) return false;
    return buscaBinariaRec (x, raiz);
  }
  boolean buscaBinariaRec (int x, No atual) {
    if (atual == null) return false;
    if (atual.getInfo() == x) return true;
    if (x < atual.getInfo()) return buscaBinariaRec(x, atual.getEsquerda());
    else return buscaBinariaRec(x, atual.getDireita());
  }
```

Desenvolva um método para essa classe que encontra o nível de um nó que contém um valor x (passado como parâmetro). Caso a árvore esteja vazia ou o valor não for encontrado, o retorno é -1. **Observação**: pode ser necessário realizar o método em 2 etapas.

Resposta

4. Um dos algoritmos de ordenação estudados foi o Quicksort. Sabemos que esse algoritmo pode ser implementado de forma recursiva e que a estratégia utilizada é a divisão e conquista, a mesma utilizada no algoritmo Mergesort. A alma do Quick é o particionamento, que consiste em, a partir de um pivô escolhido, levar os menores ou iguais a ele para sua esquerda e os maiores para sua direita. O particionamento não ordena a lista de dados, apenas o pivô daquela rodada está na posição certa. O algoritmo do particionamento é dado a seguir e este código está considerando um vetor v de valores numéricos.

```
int partition (int p, int r) {
  double x = v[r];
  int i = p-1;
  for (int j = p; j < r; j++) {
    if (v[j] <= x) {
      i = i + 1;
      double aux = v[i];
      v[i] = v[j];
      v[j] = aux;
    }
  }
  i = i + 1;
  v[r] = v[i];
  v[i] = x;
  return i;
}
```

Responda às questões a seguir **direta, suscinta e objetivamente**:

a) o que é a variável p? (0,5)

b) o que é a variável r? (0,5)

c) qual variável é o pivô e qual sua posição no início do algoritmo? (1,0)

d) o que é o retorno i? (0,5)