Questão 1 :

```java
import java.util.*;

public class Teste {

    public static void quicksort(int[] array, int esq, int dir) {

        int comp = 0, mov = 0;

        int i = esq, j = dir, pivo = array[(esq + dir) / 2];
        while (i <= j) {
            while (array[i] < pivo)
                i++;
            while (array[j] > pivo)
                j--;
            if (i <= j) {
                trocar(array, i, j);
                i++;
                j--;
            }
        }
        if (esq < j)
            quicksort(array, esq, j);
        if (i < dir)
            quicksort(array, i, dir);

        System.out.println("Comparações " + comp);
        System.out.println("Movimentações " + mov);
    }

    public static void trocar(int array[], int i, int j) {

        int temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int n;
        int vet[];

        System.out.println("Digite o tamanho de Vetor");
        n = sc.nextInt();
```

```
        vet = new int[n];

        System.out.println("Preencha o Vetor");
        for (int i = 0; i < vet.length; i++) {
            vet[i] = sc.nextInt();
        }

        quicksort(vet, 0, n - 1);

        System.out.println("Vetor Ordenado");
        for (int i = 0; i < vet.length; i++) {
            System.out.print(" " + vet[i]);
        }

        sc.close();

    }
}
```

Questão 2:

```
package src;

import java.util.*;

public class Teste {

    public static void mergesort(int[] array, int esq, int dir) {
        if (esq < dir) {
            int meio = (esq + dir) / 2;
            mergesort(array, esq, meio);
            mergesort(array, meio + 1, dir);
            intercalar(array, esq, meio, dir);
        }
    }

    public static void intercalar(int[] array, int esq, int meio, int
dir) {
        // Definir tamanho dos dois subarrays
        int nEsq = meio - esq + 1;
        int nDir = dir - meio;
        int[] arrayEsq = new int[nEsq + 1];
        int[] arrayDir = new int[nDir + 1];
        // Sentinela no final dos dois arrays
        arrayEsq[nEsq] = Integer.MAX_VALUE;
        arrayDir[nDir] = Integer.MAX_VALUE;
```

```
        int iEsq, iDir, i;
        // Inicializar primeiro subarray

        for (iEsq = 0; iEsq < nEsq; iEsq++) {
            arrayEsq[iEsq] = array[esq + iEsq];
        }
        // Inicializar segundo subarray
        for (iDir = 0; iDir < nDir; iDir++) {
            arrayDir[iDir] = array[(meio + 1) + iDir];
        }
        // Intercalacao propriamente dita
        for (iEsq = 0, iDir = 0, i = esq; i <= dir; i++) {
            array[i] = (arrayEsq[iEsq] <= arrayDir[iDir]) ?
arrayEsq[iEsq++] : arrayDir[iDir++];
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int n;
        int vet[];

        System.out.println("Digite o tamanho de Vetor");
        n = sc.nextInt();

        vet = new int[n];

        System.out.println("Preencha o Vetor");
        for (int i = 0; i < vet.length; i++) {
            vet[i] = sc.nextInt();
        }

        mergesort(vet, 0, n - 1);

        System.out.println("Vetor Ordenado");
        for (int i = 0; i < vet.length; i++) {
            System.out.print(" " + vet[i]);
        }

        sc.close();

    }
}
```

Questão 3 :

3) Ordene o vetor [10, 1, 3, 20, 5, 6, 1, 4, 9, 2]

a) Quicksort:

Vetor Original:

[10, 1, 3, 20, 5, 6, 1, 4, 9, 2]

[10, 1, 3, 20, 5, 6, 1, 4, 9, 2] – 5: pivô

[2, 1, 3, 4, 1, 6, 5, 20, 9, 10] – 3: pivô do subvetor

[2, 1, 1, 4, 3, 6, 5, 20, 9, 10] – 1: pivô do subvetor do subvetor

[1, 1, 2, 4, 3, 6, 5, 20, 9, 10] – 4: pivô da outra metade do primeiro subvetor

[1, 1, 2, 3, 4, 6, 5, 20, 9, 10] – 20: pivô do subvetor

[1, 1, 2, 3, 4, 6, 5, 10, 9, 20] – 5: pivô do subvetor do subvetor

[1, 1, 2, 3, 4, 5, 6, 10, 9, 20] – 10: pivô da outra metade do primeiro subvetor

Vetor Ordenado:

[1 - 1 - 2 - 3 - 4 - 5 - 6 - 9 - 10 - 20]

b) Mergesort:

[10, 1, 3, 20, 5, 6, 1, 4, 9, 2] - Vetor Original;

[10,1,3,20,5] - [6,1,4,9,2] //Separando em dois subvetores;

[10,1,3][20,5]-//dividindo o subvetor da esquerda em 2;

[10,1][3]- //dividindo o subvetor do subvetor;

[10][1]- //separando o elemento em um vetor único e ordenado para comparar com o outro elemento;

[1,10][3]- //ordenando de forma intercalada;

[1,3,10] - [20,5] //subvetor do subvetor ordenado, agora repete o processo com o outro subvetor do subvetor;

- [20] [5] //separando os elementos em um vetor único e ordenado para comparar com o outro elemento;

[1,3,10] - [5,20] //ambos subvetores da metade esquerda do vetor original estão ordenados, agora vamos unificá-los em uma única metade do vetor já ordenada;

[1,3,5,10.20]//metade ordenada - [6,1,4,9,2] //outra metade do vetor aguardando execução;

- [6,1,4]-[9,2]//dividindo a metade direita do vetor em 2 subvetores;

- [6,1][4]-//dividindo o subvetor gerado em outro subvetor;

- [6][1]// separando os elementos para ordenar de forma intercalada;

- [1,6][4] - //ordenando de forma intercalada;

- [1,4,6]-[9,2] //metade do subvetor da direita foi organizado, agora vamos ordenar a outra metade;

-[9][2] // separando os elementos...;

-[2,9] //ordenando de forma intercalada;

[1,3,5,10,20] - [1,2,4,6,9] // ambos subvetores gerados do vetor original já estão ordenados entre si;

[1,1,2,3,5,6,9,10,20] //unificando os subvetores em um único vetor ordenado.;

Questão 4 :

Número de comparações

	QuickSort	MergeSort
Vetor crescente	54	88
Vetor decrescente	38	88
Vetor aleatório	44	88

Número de Movimentações

	QuickSort	MergeSort
Vetor crescente	36	88
Vetor decrescente	66	88
Vetor aleatório	69	88