1. questão
    a.

ArrayList

```java
import java.util.ArrayList;

public class ArrayLi {
    Run | Debug
    public static void main(String[] args) throws Exception {
        ArrayList<Integer> list = new ArrayList<>(); // ciar uma variavel tipo array list

        list.add(10); // adiciona uma novo valor no array
        list.add(20);
        list.add(30);
        list.add(40);

        for(int i = 0; i < list.size(); i++){ // metodo size retorna o tamanho total do array
            System.out.println("todos os valores"+ list.get(i) + "\n");// metodo .get pega o valor correspondente ao index
        }


        list.remove(2); //metodo .remove remove o valor armazenado no index

        for(int i = 0; i < list.size(); i++){
            System.out.println("remoção "+list.get(i) + "\n");
        }
    }
}
```

LinkediList

```java
import java.util.LinkedList;

public class LinkedLi {
    Run | Debug
    public static void main(String[] args) throws Exception {
        LinkedList<Integer> list = new LinkedList<>(); // declara um variavel tipo LinkedList

        list.add(10); //adiciona novo elemento na lista
        list.add(20);
        list.add(30);
        list.add(40);

        for(int i = 0; i < list.size(); i++){ // metodo size retorna o tamanho total do array
            System.out.println("todos os valores"+ list.get(i) + "\n");// metodo .get pega o valor correspondente ao index
        }

        list.remove(1); //remove o elemento armazenado no index da lista

        for(int i = 0; i < list.size(); i++){ // metodo size retorna o tamanho total do array
            System.out.println("remoção "+ list.get(i) + "\n");// metodo .get pega o valor correspondente ao index do array
        }
    }
}
```

b. *ArrayList*: quando o arrayList chega ao limite do seu tamanho, mas ainda tem há valores que precisam ser inseridos no array, o arrayList faz uma cópia do arrayList origina com 50% do tamanho do arrayList original

LinkeList: quando o LinkeList chega ao limite do seu tamanho, é criado uma Double linked list

c. ArrayList: o custo de aumento do tamanho do array quando a capacidade total é preenchida é muito alto,
   LinkedList: o linkedList possui uma perfomace maior nos metodos add é remove em comparação ao arrayList, porem os metodos get é set possuem uma performace pior que a do arrayList

2. ArrayList 10.000 elementos: 52 Milissegundos
   ArrayList 100;000 elemenstos: 1461 Milissegundos
   ArrayList 1.000.000 elementos: 158533 Milissegundos
   LinkedList 10.000 elementos: 50 Milissegundos
   LinkedList 100.000 elementos: 883 Milissegundos
   LinkedList 1.000.000 elementos: 140824 Milissegundos
3. Pelo fato de o ArrayList criar uma cópia do dele mesmo quando atinge a sua capacidade máxima tem um grande efeito sobre a performasse do programa, na primeira execução tanto o tempo do ArrayList e o tempo do LinkedList tem uma diferença de 2 Milissegundos, porém à medida que o número de elementos inserido no array começo a aumentar a diferença do tempo de cada um começo a fica maior.
4. Definir um tamanho fixo para o ArrayList, assim não vai ter a necessidade de aumentar o tamanho cada vez que fica com a capacidade máxima