Q. 1) What is the difference between shallow copy and deep copy in collections?

Shallow Copy:

A shallow copy creates a new collection object but does not create new copies of the objects inside it. Instead, it copies only the references of the objects. Both collections point to the same objects in memory.

Example:

```
ArrayList list1 = new ArrayList<>();
 list1.add(new Student("mahesh", 20));
 ArrayList list2 = new ArrayList<>(list1);
```

In this case, list1 and list2 are different lists, but they contain references to the same Student object. If you modify the object using list2, the change will reflect in list1 as well.

Deep copy:

A deep copy creates a new collection and also creates new copies of all objects inside it. Each object is independently recreated.

```
Example: ArrayList list2 = new ArrayList<>();
 for(Student s : list1){
        list2.add(new Student(s.getName(), s.getAge()));
}
```

Q. 2) How Do You Sort a Collection?

Sorting in Java can be done in multiple ways.

A) Using Comparable (Natural Ordering)

If a class implements Comparable, it defines its natural sorting order.

```
class Student implements Comparable {
      int age;
```

```
        public int compareTo(Student s) {
         return this.age - s.age;
    }
}
```

Collections.sort(list);

B) Using Comparator (Custom Sorting)

Used when we want custom sorting logic or multiple sorting conditions.

Collections.sort(list, (a, b) -> a.getName().compareTo(b.getName()));

C) Using List.sort() (Java 8+)

list.sort((a, b) -> a - b);

D) Using Stream API

list.stream().sorted().forEach(System.out::println);

Q. 3) How Do You Convert an Array to a List?

Using Arrays.asList():

String[] arr = {"A", "B", "C"};

List list = Arrays.asList(arr);

Important:

- The returned list is fixed-size.

- You cannot add or remove elements.

- It is backed by the original array.

To create a modifiable list:

List list = new ArrayList<>(Arrays.asList(arr));

Special Case (Primitive Arrays):

int[] arr = {1,2,3};

Arrays.asList(arr);

Use Integer[] instead of int[].

Q. 4) How Do You Convert a List to an Array?

Using toArray():

```
List list = new ArrayList<>();

list.add("Java");

String[] arr = list.toArray(new String[0]);
```

The parameter new String[0] allows Java to create an array of the correct size.

For primitive array conversion:

```
List list = Arrays.asList(1,2,3);

int[] arr = list.stream().mapToInt(i -> i).toArray();
```

Q. 5) What Are Lambda Expressions?

Lambda expressions are a feature introduced in Java 8 that enable functional programming style in Java

- Can access effectively final local variables.
- Reduces boilerplate code.
- Enables behavior parameterization.
- Improves readability for small logic blocks.
- Less code
- Better readability
- Enables parallel processing (Streams)
- Encourages functional programming
- Works only with functional interfaces
- Not suitable for complex multi-line logic
- Can reduce readability if overused.

Q. 6) Write a program to find common keys between two HashMaps.

```
import java.util.*;

public class CommonKeysExample {

    public static void main(String[] args) {

        HashMap<Integer, String> map1 = new HashMap<>();

        map1.put(1, "Java");
        map1.put(2, "Python");
        map1.put(3, "C++");

        HashMap<Integer, String> map2 = new HashMap<>();

        map2.put(2, "Python");
        map2.put(3, "C++");
        map2.put(4, "JavaScript");

        Set<Integer> commonKeys = new HashSet<>(map1.keySet());
        commonKeys.retainAll(map2.keySet());

        System.out.println("Common Keys: " + commonKeys);
    }
}
```

Q. 7) Program to Remove a Key from HashMap

```
import java.util.*;

public class RemoveKeyExample {

    public static void main(String[] args) {

        HashMap<Integer, String> map = new HashMap<>();
        map.put(1, "Java");
        map.put(2, "Python");
        map.put(3, "C++");

        map.remove(2);
        System.out.println("After removing key 2: " + map);
    }
}
```

Q. 8) Program to Check if a Key Exists in HashMap

```java
import java.util.*;

public class CheckKeyExample {

    public static void main(String[] args) {

        HashMap<Integer, String> map = new HashMap<>();
        map.put(1, "Java");
        map.put(2, "Python");

        if (map.containsKey(1)) {
            System.out.println("Key 1 exists");
        } else {
            System.out.println("Key 1 does not exist");
        }

    }
}
```