1) Write a java program to display permutation given string?

```
 Input:
 ABC
 output:
 ABC
 ACB
 BAC
 BCA
 CBA
 CAB
```

sol:
=====

```java
public class Test {

    public static String swap(String str, int i, int j) {
        char[] charArray = str.toCharArray();
        char temp = charArray[i];
        charArray[i] = charArray[j];
        charArray[j] = temp;
        return String.valueOf(charArray);
    }

    public static void permute(String str, int start, int end) {
        if (start == end) {
            System.out.println(str);
        } else {
            for (int i = start; i <= end; i++) {
                str = swap(str, start, i);
                permute(str, start + 1, end);
                str = swap(str, start, i);
            }
        }
    }

    public static void main(String[] args) {
        String input = "ABC";
        int n = input.length();
        System.out.println("Permutations of the string \"" + input + "\":");
        permute(input, 0, n - 1);
    }
}
```

2) Write a java program to perform multiplication of two arrays?

```
 input:
 arr1 = 6 2 4
 arr2 = 3 1
 output:
 19344(624*31)
```

sol:
=====

```
public class Test {
        public static void main(String[] args) {
                int[] arr1 = { 6, 2, 4 };
                int[] arr2 = { 3, 1 };

                int rem1 = 0;
                for (int i : arr1) {
                        rem1 *= 10;
                        rem1= rem1 + i;
                }
                int rem2 = 0;
                for (int i : arr2) {
                        rem2 *= 10;
                        rem2 = rem2 + i;
                }
                System.out.println(rem1);
                System.out.println(rem2);
                System.out.println(rem1*rem2);

        }
}
```

3) Write a java program to display employee details based on sorting order of salary by
using collections?

sol:
=====

```
package mypack;
import java.util.*;

class Employee {
        int empId;
        String empName;
        int empAge;
        int salary;

        public Employee() {

        }

        public Employee(int empId, String empName, int empAge, int salary) {
                super();
                this.empId = empId;
                this.empName = empName;
                this.empAge = empAge;
                this.salary = salary;
        }

        @Override
        public String toString() {
                return "Employee [empId=" + empId + ", empName=" + empName + ", 
empAge=" + empAge + ", salary=" + salary + "]";
        }
}

class IdCompare implements Comparator<Employee> {
        @Override
        public int compare(Employee o1, Employee o2) {
                if(o1.salary>o2.salary)
                        return 90;
                else if(o1.salary < o2.salary)
                        return -90;
                return 0;
        }
}

public class Test {
        public static void main(String[] ar) {

                TreeMap<Employee, Integer> lhm = new TreeMap(new IdCompare());
                Employee emp1 = new Employee(123, "swarna raj", 21, 900000);
                Employee emp2 = new Employee(124, "sandeep", 22, 120002);
                Employee emp3 = new Employee(124, "sai", 21, 65400);
                Employee emp4 = new Employee(125, "Hari", 22, 902002);
                Employee emp5 = new Employee(126, "rahul", 23, 89020);

                lhm.put(emp1, 3);
                lhm.put(emp2, 2);
                lhm.put(emp3, 6);
                lhm.put(emp4, 5);
                lhm.put(emp5, 9);

                for (Map.Entry<Employee, Integer> s : lhm.entrySet())
                        System.out.println(s.getKey() + " " + s.getValue());
        }
}
```

4) Write a Java program to check given string is well formed/Balanced or
not by using collections?
Input:
 ([{}])
 Output:
 Balanced

sol:
=====

```
import java.util.Stack;

public class Test {

    public static boolean isBalanced(String str) {
        Stack<Character> stack = new Stack<>();
        for (int i = 0; i < str.length(); i++) {
            char currentChar = str.charAt(i);

            if (currentChar == '(' || currentChar == '{' || currentChar == '[')
{
                stack.push(currentChar);
            }
            else if (currentChar == ')' || currentChar == '}' || currentChar ==
']') {
                if (stack.isEmpty()) {
                    return false;
                }
                char topChar = stack.pop();
                if (!isMatchingPair(topChar, currentChar)) {
                    return false;
                }
            }
        }

        return stack.isEmpty();
    }

    public static boolean isMatchingPair(char open, char close) {
        return (open == '(' && close == ')') ||
               (open == '{' && close == '}') ||
               (open == '[' && close == ']');
    }

    public static void main(String[] args) {
        String input = "([{}])";

        if (isBalanced(input)) {
            System.out.println("Balanced");
        } else {
            System.out.println("Not Balanced");
        }
    }
}
```