Core Java Day-10

**Q.**Which of the given syntax is correct to display elements of an array given below using streams?  
String[]strng = new String[]{"IGATE","GLOBAL","SOLUTION"};

**1)**. **Stream<String>stream = Arrays.stream(strng);  
stream.forEach((str)->{  
System.out.println(str);});**

**2)**. Stream<String>stream = Arrays.stream(strng);  
stream.forEach((str)->  
System.out.println(str););

**3)**. Stream<String>stream = Arrays.getStream(strng);  
stream.forEach((str)->  
System.out.println(str););

**4)**. Stream<String>stream = Arrays.getStream(strng);  
stream.forEach((str)->{  
System.out.println(str);});

**Q.** String []strng = new String[]{"abc","xyz","abcxyz"};  
Stream<String>stream = Arrays.stream(strng);  
System.out.println(stream.skip(1).findFirst());  
What will be the output of above code?

**1)**. abc

**2)**. abcxyz

**3)**. xyz

**4)**. **Optional[xyz]**

**Q.** Which of the given are characteristics of Stream API?

**1)**. **Designed for lamdas**

**2)**. **Can easily be output as an array or list**

**3)**. **Support indexed access**

**4)**. **bounded**

**Q.** String []strng = new String[]{"abc","xyz","abcxyz"};  
Stream<String>stream = Arrays.stream(strng);  
System.out.println(stream.limit(1).findAny());  
What will be the output of above code?

**1)**. abc

**2)**. xyz

**3)**. **Optional[abc]**

**4)**. Optional[xyz]

**Q.** @FunctionalInterface StringI  
{  
String getString();  
}Which of the given syntax is correct to call this getString() method?

**1)**. ("Welcome to India")->getString();

**2)**. **StringI ref =()->"Welcome to India";  
  System.out.println(ref.getString());**

**3)**. String ref =("Welcome to India")->getString();  
  System.out.println(ref);

**4)**. none of the above

**Q.** String []strng = new String[]{"abc","xyz","abcxyz"};  
Select the correct option to print only the string elements that contains "xyz"?

**1)**. **Stream<String>stream = Arrays.stream(strng);  
  stream.filter(str->str.contains("xyz")).forEach(strn->System.out.println(strn));**

**2)**. Stream<String>stream = Arrays.stream(strng);  
  stream.distinct(str->str.contains("xyz")).forEach(strn->System.out.println(strn));

**3)**. System.out.println(stream.allMatch(str1->str1.contains("xyz")));

**4)**. System.out.println(stream.anyMatch(str1->str1.contains("xyz")));

**Q.** @FunctionalInterface  
interface Power  
{  
 public int calculatePower(int base,int index);  
}  
public class CalculatePowerDemo{  
  
 public static void main(String[] args) {  
int x=4,y=2;  
}  
}How to call calculatePower() in CalculatePowerDemo class?

**1)**. Class CalculatePowerDemo need to implement Power interface and has to override calculatePower() to call it

**2)**. **It can be implemented as given below  
Power pwr = (base,index)->{for(int i=0;i<index;i++)  
  {  
    base = base\*index;  
  }  
  return base;};  
  result= pwr.calculatePower(x, y);  
  System.out.println(result);**

**3)**. It can be implemented as given below  
Power pwr = (base,index)->{for(int i=0;i<index;i++)  
  {  
    base = base\*index;  
  }  
  };  
  result= pwr.calculatePower(x, y);  
  System.out.println(result);

**4)**. It can be implemented as given below  
System.out.println((base,index)->{for(int i=0;i<index;i++)  
  {  
    base = base\*index;  
  }  
  return base;};)

**Q.** BiFunction<Integer,Integer,Integer>max = (x,y)->(x>y)?x:y;  
  System.out.println(max.apply(5, 52));  
Which of the given statements are true about above code?

**1)**. **It generate the output as 50**

**2)**. Compilation error, as apply need to be called with 3 arguments

**3)**. It will execute successfully if below change is done  
BiFunction<Integer,Integer>max = (x,y)->(x>y)?x:y;

**4)**. **Same expression can also be written as  
BinaryOperator<Integer> maxFunction = (x,y) -> x>y?x:y;  
System.out.println(max.apply(5,52));**

**Q.** Which of the given options are built-in functional interfaces?

**1)**. **supplier**

**2)**. **consumer**

**3)**. **predicate**

**4)**. producer

**Q.** Which of the given statements are true about functional interface?

**1)**. **Use of @FunctionalInterface is optional**

**2)**. **It has exactly one abstract method**

**3)**. **It can also have static and default method**

**4)**. It may contain more than one abstract method