1. What will be the output of the following code?

```
public class Main {
    public static void fun(int n) {
        if (n == 0) return;
        System.out.print(n + " ");
        fun(n - 1);
    }
    public static void main(String[] args) {
        fun(5);
    }
}
```

A) 1 2 3 4 5

B) 5 4 3 2 1

C) 5 4 3 2

D) Infinite loop

Answer: B) 5 4 3 2 1.

Explanation: The function prints before the recursive call, so values are printed in reverse order from 5 down to 1.

2. What will be the output of the following recursive function?

```
public class Main {
    public static void fun(int n) {
        if (n == 0) return;
        fun(n - 1);
        System.out.print(n + " ");
    }
    public static void main(String[] args) {
        fun(3);
    }
}
```

A) 3 2 1

B) 1 2 3

C) 3 2 1 0

D) Compilation error

Answer: B) 1 2 3

Explanation: Since the function prints after the recursion, it prints values in increasing order.

3. What is the base case in the following recursive function?

```
public class Main {
    public static int sum(int n) {
        if (n == 1) return 1;
        return n + sum(n - 1);
    }
    public static void main(String[] args) {
        System.out.println(sum(5));
    }
}
```

A) if (n == 0) return 0;

B) if (n == 1) return 1;

C) return n + sum(n - 1);

D) No base case

Answer: B) if (n == 1) return 1;

Explanation: The base case is the condition that stops recursion; here it stops when n is 1.

4. What will be the output of the following recursive function?

```
public class Main {
    public static int factorial(int n) {
        if (n == 0) return 1;
        return n * factorial(n - 1);
    }
    public static void main(String[] args) {
        System.out.println(factorial(4));
    }
}
```

A) 24

B) 10

C) 4

D) Compilation error

Answer: A) 24

Explanation: The factorial is calculated as 4 × 3 × 2 × 1 = 24.

5. What is the output of the following recursive function?

```
public class Main {
    public static int fun(int n) {
        if (n <= 1) return n;
        return fun(n - 1) + fun(n - 2);
    }
    public static void main(String[] args) {
        System.out.println(fun(5));
    }
}
```

A) 5

B) 8

C) 10

D) 12

Answer: A) 5

Explanation: The 5th Fibonacci number (0-based) is 5 (0, 1, 1, 2, 3, 5).

6. What will be the output of the following recursive function?

```
public class Main {
    public static void fun(int n) {
        if (n == 0) return;
        System.out.print(n + " ");
        fun(n / 2);
    }
    public static void main(String[] args) {
        fun(10);
    }
}
```

A) 10 5 2 1
B) 10 5 2 1 0
C) 10 5 2
D) 10 2 1
Answer: A) 10 5 2 1
Explanation: n is halved each time until 0, printing values before each recursive call.

7. What will be the output of the following recursive function?

```
public class Main {
    public static void fun(int n) {
        if (n <= 0) return;
        fun(n - 1);
        System.out.print(n + " ");
        fun(n - 1);
    }
    public static void main(String[] args) {
        fun(3);
    }
}
```

A) 3 2 1
B) 1 2 1 3 1 2 1
C) 1 2 3 2 1
D) Compilation error
Answer: B) 1 2 1 3 1 2 1
Explanation: This creates a binary tree of recursive calls and prints in between them.

8. What is the output of the following function?

```
public class Main {
    public static int fun(int n) {
        if (n == 1) return 1;
        return 2 * fun(n - 1);
    }
    public static void main(String[] args) {
        System.out.println(fun(4));
    }
}
```

A) 8
B) 16
C) 4
D) 32
Answer: A) 8
Explanation: 2 × 2 × 2 = 8, because fun(1) returns 1 and we double it three times.

9. What will be the output of this function?

```
public class Main {
    public static void print(int n) {
        if (n == 0) return;
        System.out.print(n + " ");
        print(n - 1);
        System.out.print(n + " ");
    }
    public static void main(String[] args) {
        print(3);
    }
}
```

A) 3 2 1 1 2 3
B) 3 2 1
C) 1 2 3
D) 3 2 1 2 3
Answer: A) 3 2 1 1 2 3
Explanation: Prints values before and after recursion, creating a mirrored output.
10. What will be the output of the following function?

```
public class Main {
    public static int power(int base, int exp) {
        if (exp == 0) return 1;
        return base * power(base, exp - 1);
    }
    public static void main(String[] args) {
        System.out.println(power(2, 3));
    }
}
```

A) 8
B) 6

C) 9

D) 12

Answer: A) 8.

Explanation: 2^3 = 2 × 2 × 2 = 8 using recursive multiplication.

11. What is the recurrence relation for the Fibonacci sequence?

a) F(n) = F(n-1) + F(n-2)

b) F(n) = F(n-1) * F(n-2)

c) F(n) = F(n-1) - F(n-2)

d) F(n) = F(n-1) / F(n-2)

Answer: a) F(n) = F(n-1) + F(n-2).

Explanation: Each number is the sum of the two previous numbers.

12. What is the base case in the recursive factorial function?

a) n == 1

b) n == 0

c) n == -1

d) No base case

Answer: b) n == 0

Explanation: By definition, 0! = 1; this stops the recursion.

13. What will be the output of the following Java program?

```
class recursion
{
    int func (int n)
    {
        int result;
        result = func (n - 1);
        return result;
    }
}
class Output
{
    public static void main(String args[])
    {
        recursion obj = new recursion() ;
        System.out.print(obj.func(12));
    }
}
```

a) 0

b) 1

c) Compilation Error

d) Runtime Error

Answer: d

Explanation: Since the base case of the recursive function func() is not defined hence infinite loop occurs and results in Stack

Overflow.

Output:

```
javac Output.javac
java Output
Exception in thread "main" java.lang.StackOverflowError
```

14. What will be the output of the following Java program?

```
class recursion
{
    int func (int n)
    {
        int result;
        if (n == 1)
            return 1;
        result = func (n - 1);
        return result;
         }
      }
      class Output
      {
          public static void main(String args[])
          {
              recursion obj = new recursion() ;
              System.out.print(obj.func(5));
          }
      }
```

a) 0

b) 1

c) 120

d) None of the mentioned

Answer: b

Explanation: The function recursively calls itself until n equals 1, then returns 1. Since there’s no operation besides returning the recursive call’s result, the final output is always 1 regardless of the input.

Output:

```
javac Output.javac
java Output
1
```

15. What will be the output of the following Java program?

```
class recursion
{
    int fact(int n)
    {
        int result;
        if (n == 1)
            return 1;
        result = fact(n - 1) * n;
        return result;
          }
      }
      class Output
      {
          public static void main(String args[])
          {
              recursion obj = new recursion() ;
              System.out.print(obj.fact(5));
          }
      }
```

a) 24
b) 30
c) 120
d) 720
Answer: c
Explanation: fact() method recursively calculates factorial of a number, when value of n reaches 1, base case is executed and 1 is returned. In each call the expression 5*4*3*2*1 is formed to stack the final call folded and return the result as their product i.e. 120.
Output:

```
javac Output.javac
java Output
120
```

16. What will be the output of the following Java program?

```
class recursion
{
    int fact(int n)
    {
        int result;
        if (n == 1)
            return 1;
        result = fact(n - 1) * n;
        return result;
          }
      }
      class Output
      {
```

```
    public static void main(String args[])
    {
        recursion obj = new recursion() ;
        System.out.print(obj.fact(1));
    }
}
```

a) 1
b) 30
c) 120
d) Runtime Error
Answer: a
Explanation: fact() method recursively calculates factorial of a number, when value of n reaches 1, base case is excuted and 1 is returned.
Output:

```
javac Output.javac
java Output
1
```

17. What will be the output of the following Java program?

```
class recursion
{
    int fact(int n)
    {
        int result;
        if (n == 1)
            return 1;
        result = fact(n - 1) * n;
        return result;
         }
     }
     class Output
     {
         public static void main(String args[])
         {
             recursion obj = new recursion() ;
             System.out.print(obj.fact(6));
         }
     }
```

a) 1
b) 30
c) 120
d) 720

Answer: d
Explanation: fact() method recursively calculates factorial of a number, when value of n reaches 1, base case is executed and 1 is returned. In each call the expression 6*5*4*3*2*1 is formed to stack the final call folded and return the result as their product i.e. 720.
Output:

```
javac Output.javac
java Output
720
```

Q18:-1Which line has the recursive call?

```
1 public String starString(int n)
2 {
3    if (n == 0) {
4       return "*";
5    } else {
6       return starString(n - 1) + starString(n - 1);
7    }
8 }
```

A. 1
B. 3
C. 4
D. 5
E. 6
Answer: E. Line 6
Explanation:
Line 6 contains the recursive call: return starString(n - 1) + starString(n - 1);

Q19:-2How many recursive calls does the following method contain?

```
1 public static int fibonacci(int n)
2 {
3    if (n == 0)
4       return 0;
5    else if (n == 1)
6       return 1;
7    else return fibonacci(n-1) + fibonacci(n-2);
8     }
```

A. 0
B. 1

C. 2

D. 3

Answer: C. 2

Explanation:

When n > 1, it calls fibonacci(n-1) and fibonacci(n-2), hence 2 recursive calls.

Q:20- How many recursive calls does the following method contain?

```
public static int multiplyEvens(int n)
{
   if (n == 1) {
      return 2;
   } else {
      return 2 * n * multiplyEvens(n - 1);
   }
}
```

A. 0

B. 1

C. 2

D. 3

Answer: B. 1

Explanation:

There is one recursive call inside the else block: multiplyEvens(n - 1).

Q21:- Given the following method declaration, which of the following is printed as the result of the call mystery(1234)?

```
//precondition:  x >=0
public static void mystery (int x)
{
   System.out.print(x % 10);

   if ((x / 10) != 0)
   {
      mystery(x / 10);
   }
   System.out.print(x % 10);
}
```

A. 1441

B. 43211234

C. 3443

D. 12344321

E. Many digits are printed due to infinite recursion.

Check MeCompare me

Answer: B. 43211234

Explanation:

The digits are printed twice – once on the way down and once on the way back from recursion.

- First prints: 4, then 3, 2, 1
- Then prints again in reverse: 1, 2, 3, 4
  Total: 43211234

Q22:- Given the following method declaration, what value is returned as the result of the call mystery(5)?

```
public static int mystery(int n)
{
   if (n == 0)
      return 1;
   else
      return 3 * mystery (n - 1);
}
```

A. 243

B. 0

C. 3

D. 81

E. 27

Answer: A. 243

Explanation:

This computes 3^5 = 243, since the recursive call multiplies by 3 for each n.

Q23:- Given the following method declaration, what value is returned as the result of the call product(5)?

```
public static int product(int n)
{
   if (n <= 1)
      return 1;
   else
      return n * product(n - 2);
}
```

A. 1
B. 10
C. 25
D. 3125
E. 15
Answer: E. 15
Explanation:
Calls: product(5) → 5 * product(3) → 5 * 3 * product(1) → 5 * 3 * 1 = 15

Q24:- Given the following method declaration, what value is returned as the result of the call f(5)?

```
public static int f(int n)
{
   if (n == 0)
      return 0;
   else if (n == 1)
      return 1;
   else return f(n-1) + f(n-2);
}
```

A. 8
B. 3
C. There is no result because of infinite recursion.
D. 5
E. 0
Answer: A. 5
Explanation:
This is the Fibonacci function, so:
f(5) = f(4) + f(3) = 3 + 2 = 5

Q25:- Given the following method declaration, this method will return true if and only if:

```
public static boolean check(String s)
{
   return s.length() >= 2 &&
          (s.charAt(0) == s.charAt(1) ||
           check(s.substring(1)));
}
```

A. The string s contains two or more of the same characters.
B. The string s starts with two or more of the same characters.
C. The string s contains two or more of the same character that are next to each other.

D. The string s ends with two or more of the same characters

Answer: C. The string s contains two or more of the same character that are next to each other.

Explanation:

The method returns true if two adjacent characters are the same anywhere in the string.

Q26:- Given the following method declaration, what will redo(82, 3) return?

```
public static int redo(int i, int j)
{
   if (i==0)
      return 0;
   else
      return redo(i/j, j)+1;
}
```

A. 5

B. 4

C. 6

D. 7

E. The method never returns due to infinite recursion.

Answer: A. 5

Explanation:

Steps:

- redo(82, 3) → redo(27,3) +1
- redo(27,3) → redo(9,3) +1
- redo(9,3) → redo(3,3) +1
- redo(3,3) → redo(1,3) +1
- redo(1,3) → redo(0,3) +1 → base case

Total calls = 5 → returns 5

Question 27:-

Predict output of following program

```
public class Main {
    public static int fun(int n) {
        if (n == 4)
            return n;
        else return 2 * fun(n + 1);
    }
    public static void main(String[] args) {
        System.out.println(fun(2));
    }
}
```

A 4

B 8

C 16

D Runtime Error

Answer: C) 16

Explanation:

Recursive calls:

- fun(2) → 2 * fun(3)
- fun(3) → 2 * fun(4)
- fun(4) → returns 4

  Then backtrack: 2*4 = 8, 2*8 = 16

Question 28:-

Consider the following recursive function fun(x, y). What is the value of fun(4, 3)

```
int fun(int x, int y) {
  if (x == 0)
    return y;
  return fun(x - 1, x + y);
}
```

A 13

B 12

C 9

D 10

Answer: A) 13

Explanation:

fun(4,3) → fun(3,7) → fun(2,10) → fun(1,12) → fun(0,13) = 13

Question 29:-

What does the following function print for n = 25?

```
public class Main {
    public static void fun(int n) {
        if (n == 0)
            return;
        System.out.print(n % 2);
        fun(n / 2);
    }
    public static void main(String[] args) {
        fun(10); // Example call
    }
}
```

A 11001

B 10011

C 11111

D 00000

Answer: B) 10011

Explanation:

Binary of 10 is 1010. Function prints bits in reverse order: 0 1 0 1 → 10011

Question 30:-

What does the following function do?

```
int fun(int x, int y)
{
    if (y == 0)   return 0;
    return (x + fun(x, y-1));
}
```

A x + y

B x + x*y

C x*y

D x^y

Answer: C) x*y

Explanation:

It adds x y times: recursive multiplication.

Question 31:-

What does fun2() do in general?

```
public class Main {
    public static int fun(int x, int y) {
        if (y == 0) return 0;
        return (x + fun(x, y - 1));
    }
    public static int fun2(int a, int b) {
        if (b == 0) return 1;
        return fun(a, fun2(a, b - 1));
    }
    public static void main(String[] args) {
        // Example usage
    }
}
```

A x*y

B x+x*y

C x^y

D y^x

Answer: C) a^b (a raised to power b)

Explanation:

It calls fun(a, fun2(a, b-1)) → repeated multiplication. Exponential recursion.

32. What is the time complexity of printing an array recursively?

a) O(1)
b) O(n)
c) O(n²)
d) O(log n)

Answer: b) O(n)

Explanation: Each recursive call processes one element → total n calls.

---

33. What will be the result of the following function call?

java
CopyEdit

```
public static void countDown(int n) {
    if (n == 0) return;
    System.out.print(n + " ");
    countDown(n - 1);
}

public static void main(String[] args) {
    countDown(3);
}
```

a) 3 2 1
b) 1 2 3
c) 0 1 2 3
d) 3 2 1 0

Answer: a) 3 2 1

Explanation: It prints and then recurses on n - 1 till 0.

34. Which case would cause a StackOverflowError in recursion?

a) Base condition missing
b) Infinite loop
c) Too many print statements
d) Using arrays with size 0

Answer: a) Base condition missing

Explanation: Without a base case, recursion never stops → infinite stack calls.

35. What is the purpose of a base case in recursive array methods?

a) To print the array
b) To end recursion
c) To call another function
d) To reset array index

Answer: b) To end recursion
Explanation: Base case ensures recursion terminates safely.

36. Which of these problems is best solved using recursion on arrays?
a) Sorting an array
b) Printing reverse order
c) Swapping two elements
d) Finding array length
Answer: b) Printing reverse order
Explanation: Printing in reverse is naturally suited for recursion.