Class 10

Chapter – 9

Nested Loop

A. TICK (✓) THE CORRECT ANSWER

Q1.

A cake has 3 layers, and each layer has 4 flavors. How many times will the inner loop execute in total?

```
for (int layer = 1; layer <= 3; layer++) {
    for (int flavor = 1; flavor <= 4; flavor++) {
        System.out.println("Layer " + layer + ", Flavor " + flavor);
    }
}
```

Options:
a. 3
b. 4
c. 7
d. 12

✓ Answer: d. 12

Explanation:

- Outer loop runs **3 times**
- Inner loop runs **4 times each**
- Total executions = **3 × 4 = 12**

Q2.

What is the final value of total number of steps?

```
int layers = 4, stepsPerLayer = 5, totalSteps = 0;

for (int i = 1; i <= layers; i++) {
    for (int j = 1; j <= stepsPerLayer; j++) {
        totalSteps++;
    }
}

System.out.println("Total baking steps: " + totalSteps);
```

Options:
a. 4
b. 5

c. 9
d. 20

✅ Answer: d. 20

💡 Explanation:

- Outer loop = 4 times
- Inner loop = 5 times
- TotalSteps = **4 × 5 = 20**

Q3.

What will be the output of the following nested for loop?

```
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print("*");
    }
    System.out.println();
}
```

Options:

a.

```
***
***
***
```

b.

```
*
**
***
```

c.

```
*
**
**
```

d.

```
***
**
*
```

✅ Answer: b

💡 Explanation:

- i=1 → *
- i=2 → **
- i=3 → ***

So triangle pattern increases.

Q4.

Given the nested while loop below, what will be the value of treasuresFound after the loop finishes if totalFloors is 2 and totalRooms is 3?

```
int totalFloors = 2;
int totalRooms = 3;
int floor = 1;
int treasuresFound = 0;

while (floor <= totalFloors) {
    int room = 1;
    while (room <= totalRooms) {
        treasuresFound++;
        room++;
    }
    floor++;
}
```

Options:
a. 2
b. 3
c. 5
d. 6

✓ Answer: d. 6

💡 Explanation:

- Floors = 2
- Rooms per floor = 3
- Total = **2 × 3 = 6**

Q5.

Each stall has 2 participants, and each participant presents 3 projects. How many times will the innermost loop execute in total?

```
for (int stall = 1; stall <= 3; stall++) {
    int participant = 1;
    while (participant <= 2) {
        int project = 1;
        do {
            System.out.println("Stall " + stall + ", Participant " + participant + ",
Project " + project);
            project++;
        } while (project <= 3);
        participant++;
```

```
    }
}
```

Options:
a. 3
b. 6
c. 9
d. 18

✓ Answer: d. 18

💡 Explanation:

- Stalls = 3
- Participants = 2
- Projects = 3

Total = **3 × 2 × 3 = 18**

B. Answer the following questions

Q1.

Write a program to input 10 numbers and print all the neon numbers.

✓ Program:

```
import java.util.Scanner;

class Neon {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        for (int i = 1; i <= 10; i++) {
            int n = sc.nextInt();
            int square = n * n;
            int sum = 0;

            while (square > 0) {
                sum += square % 10;
                square /= 10;
            }

            if (sum == n) {
                System.out.println(n + " is Neon");
            }
        }
    }
}
```

💡 Explanation:

Neon number → sum of digits of square = number
Example: 9 → 81 → 8+1=9

Q2.

Write a program to check Magic Number.

✓ Program:

```
import java.util.Scanner;

class Magic {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        while (n > 9) {
            int sum = 0;
            while (n > 0) {
                sum += n % 10;
                n /= 10;
            }
            n = sum;
        }

        if (n == 1)
            System.out.println("Magic Number");
        else
            System.out.println("Not Magic Number");
    }
}
```

Q3.

Print palindrome numbers between m and n

✓ Program:

```
import java.util.Scanner;

class PalindromeRange {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int m = sc.nextInt();
        int n = sc.nextInt();

        for (int i = m; i <= n; i++) {
            int num = i, rev = 0;

            while (num > 0) {
                rev = rev * 10 + num % 10;
                num /= 10;
```

```
            }

            if (rev == i)
                System.out.print(i + " ");
        }
    }
}
```

Q4.

Print sum of Armstrong numbers

✓ Program:

```
import java.util.Scanner;

class ArmstrongSum {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int sum = 0;

        for (int i = 1; i <= n; i++) {
            int num = i, temp = num, result = 0;

            while (temp > 0) {
                int d = temp % 10;
                result += d * d * d;
                temp /= 10;
            }

            if (result == num)
                sum += num;
        }

        System.out.println("Sum = " + sum);
    }
}
```

Q5.

Twin prime number program

✓ Program:

```
import java.util.Scanner;

class TwinPrime {
    static boolean isPrime(int n) {
        if (n < 2) return false;
        for (int i = 2; i <= n / 2; i++) {
            if (n % i == 0) return false;
        }
        return true;
    }

    public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();

        if (isPrime(a) && isPrime(b) && Math.abs(a - b) == 2)
            System.out.println("Twin Prime");
        else
            System.out.println("Not Twin Prime");
    }
}
```

Pattern Programs

Q6. Write a program to print the following patterns:

(a) Pattern

```
1
2 4
3 6 9
4 8 12 16
```

✓ Program:

```
class PatternA {
    public static void main(String[] args) {
        for (int i = 1; i <= 4; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print(i * j + " ");
            }
            System.out.println();
        }
    }
}
```

Explanation:

- Multiply row number `i` with column `j`

(b) Pattern

```
12345
2345
345
45
5
```

✓ Program:

```
class PatternB {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            for (int j = i; j <= 5; j++) {
```

```
                System.out.print(j);
            }
            System.out.println();
        }
    }
}
```

Explanation:

- Start from `i` and print till 5

(c) Pattern

```
10987
654
32
1
```

Program:

```
class PatternC {
    public static void main(String[] args) {
        int num = 10;
        for (int i = 1; i <= 4; i++) {
            for (int j = 1; j <= (5 - i); j++) {
                System.out.print(num--);
            }
            System.out.println();
        }
    }
}
```

Explanation:

- Use decreasing number (`num--`)

(d) Pattern

```
11111
2222
333
22
1
```

Program:

```
class PatternD {
    public static void main(String[] args) {
        int n = 5;

        for (int i = 1; i <= n; i++) {
            for (int j = n; j >= i; j--) {
                System.out.print(i);
            }
            System.out.println();
        }
```

```
        for (int i = n - 2; i >= 1; i--) {
            for (int j = 1; j <= i; j++) {
                System.out.print(i);
            }
            System.out.println();
        }
    }
}
```

(e) Pattern

```
1
01
101
0101
10101
```

✓ Program:

```
class PatternE {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            int num = (i % 2 == 1) ? 1 : 0;

            for (int j = 1; j <= i; j++) {
                System.out.print(num);
                num = (num == 1) ? 0 : 1;
            }
            System.out.println();
        }
    }
}
```

Q6 (f). Write a program to print the following pattern:

```
x#x#x
x#x#
x#x
x#
x
```

✓ Java Program:

```
class PatternF {
    public static void main(String[] args) {

        for (int i = 5; i >= 1; i--) {

            for (int j = 1; j <= i; j++) {

                if (j % 2 == 1)
                    System.out.print("x");
                else
                    System.out.print("#");
            }

            System.out.println();
        }
```

```
    }
}
```

💡 Explanation (Very Simple):

Step 1: Outer Loop

```
for (int i = 5; i >= 1; i--)
```

- Controls number of rows → 5 to 1

Step 2: Inner Loop

```
for (int j = 1; j <= i; j++)
```

- Controls number of characters per row

Step 3: Pattern Logic

```
if (j % 2 == 1) → print "x"
else → print "#"
```

☞ Odd positions → `x`
☞ Even positions → `#`

📌 Output:

```
x#x#x
x#x#
x#x
x#
x
```

6(f) Pattern (Simple program)

0

```
#####
####
###
##
#
```

✓ Program:

```
class PatternF {
    public static void main(String[] args) {
        for (int i = 5; i >= 1; i--) {
            for (int j = 1; j <= i; j++) {
                System.out.print("#");
            }
            System.out.println();
        }
```

```
    }
}
```

(g) Pattern

```
A
BC
DEF
GHIJ
```

✓ Program:

```
class PatternG {
    public static void main(String[] args) {
        char ch = 'A';

        for (int i = 1; i <= 4; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print(ch++);
            }
            System.out.println();
        }
    }
}
```

● C. Assertion and Reasoning Based Questions

Options:

a. Both A and R are true, and R is the correct explanation of A
b. Both A and R are true, but R is not the correct explanation of A
c. A is true, but R is false
d. A is false, but R is true

Q1

```
int i = 1;
while (i <= 3) {
    for (int j = 1; j <= 4; j++) {
        System.out.println("i: " + i + ", j: " + j);
    }
}
```

Assertion (A):

The following code will result in an infinite loop.

Reason (R):

The outer loop does not have an increment statement.

✅ Answer: a

💡 Explanation:

- `i` is never incremented → condition always true
- Loop runs forever → infinite loop

✓ Both A and R true

✓ R explains A

Q2

```
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 4; j++) {
        System.out.println("Nested Loop");
    }
}
```

Assertion (A):

The following code will print "Nested Loop" 12 times in same line.

Reason (R):

The outer loop runs 3 times, and the inner loop runs 4 times.

✅ Answer: b

💡 Explanation:

- Total prints = 3 × 4 = 12 ✓
- BUT `println` prints in **new line**, not same line ✗

✓ A is partially wrong → "same line" is incorrect
✓ R is correct

So:
👉 Both are true statements (count correct), but reason doesn't justify "same line"

****