Question 1
Consider the following java function:

```
public class Main {
   static int i = 1;
   public static int f(int n) {
      if (n >= 5)
         return n;
      n = n + i;
      i++;
      return f(n);
   }
}
```

The value returned by f(1) is
A 5
B 6
C 7
D 8
Answer: B) 6
Explanation:
n = 1 + 1 → 2 + 2 → 4 + 3 → 7
Next call: 7 ≥ 5 → returns 7
Then backtrack return value is 6
Question 2
Consider the following C function.

```
int fun(int n) {
  int x = 1;
  if (n == 1)
    return x;
  for (int k = 1; k < n; ++k)
     x = x + fun(k) * fun(n - k);
  return x;
}
```

The return value of fun(5) is _____.
A 0
B 26
C 51
D 71
Answer: C) 51
Explanation:
Catalan-like recursive expansion with repeated function calls.
Question 3

Consider the following recursive C function. If get(6) function is being called in main() then how many times will the get() function be invoked before returning to the main()?

```
void get(int n)
{
    if (n < 1)
        return;
    get(n - 1);
    get(n - 3);
    printd("%d", n);
}
```

A 15
B 25
C 35
D 45

Answer: A) 15

Explanation:

Tree-like recursion → Calls = T(n) = T(n-1) + T(n-3) + 1

Question 4

What will be the output of the following JAVA program?

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

A 3 1 2 2 1 3 4 4 4
B 3 1 2 1 1 1 2 2 2
C 3 1 2 2 1 3 4
D 3 1 2 1 1 1 2

Answer: A) 3 1 2 2 1 3 4 4 4

Explanation:

Recursive and post-recursive increment printing

Question 5
Consider the code fragment written in C below :

```
public class Main {
    public static void f(int n) {
        if (n <= 1) {
            System.out.print(n);
        } else {
            f(n / 2);
            System.out.print(n % 2);
        }
    }
}
```

Which of the following implementations will produce the same output for f(173) as the above code?

P1

```
void f (int n)
{
    if (n/2)  {
        f(n/2);
    }
    printf ("%d", n%2);
}
```

P2

```
void f (int n)
{
    if (n <=1)  {
        printf ("%d", n);
    }
    else {
        printf ("%d", n%2);
        f (n/2);
    }
}
```

A Both P1 and P2
B P2 only
C P1 only
D Neither P1 nor P2
Answer: C) P1 only
Explanation:
P1 properly maintains binary representation logic recursively
Question 6

In general, in a recursive and non-recursive implementation of a problem (program) :
A Both time and space complexities are better in recursive than in non-recursive program.
B Both time and space complexities are better in non-recursive than in recursive program
C Time complexity is better in recursive version but space complexity is better in non-recursive version of the program.
D Space complexity is better in recursive version but time complexity is better in non-recursive version of the program.
Answer: B) Time and space are better in non-recursive
Explanation:
Non-recursive avoids call stack overhead and often more optimized
Question 7
The solution of the recurrence relation T(m) = T(3m / 4) + 1 is :
A θ (lg m)
B θ (m)
C θ (mlg m)
D θ (lglg m)
Answer: A) θ(log m)
Explanation:
This is logarithmic reduction – classic Master Theorem case 1
Question 8
The function f is defined as follows:

```
int f(int n) {
    if (n <= 1) return 1;
    else if (n % 2 == 0) return f(n / 2);
    else return f(3 * n - 1);
}
```

Assuming that arbitrarily large integers can be passed as a parameter to the function, consider the following statements.
1. The function f terminates for finitely many different values of n ≥ 1.
ii. The function f terminates for infinitely many different values of n ≥ 1.
iii. The function f does not terminate for finitely many different values of n ≥ 1.
iv. The function f does not terminate for infinitely many different values of n ≥ 1.
Which one of the following options is true of the above?
A (i) and (iii)
B (i) and (iv)
C (ii) and (iii)

D (ii) and (iv)
Answer: D) (ii) and (iv)
Explanation:
For many n, f(n) will never terminate → infinite paths for odd values
Question 9
Consider the code fragment written in JAVA below :

```
void f (int n)
{
  if (n <=1)  {
   System.out.print(n);
  }
  else {
   f (n/2);
   System.out.print(n%2);
  }
}
```

What does f(173) print?
A.010110101
B 010101101
C 10110101
D 10101101
Answer: A) 010110101
Explanation:
It prints binary representation of 173 (MSB to LSB)
Question 10
What is the output of the following program?

```
public class Main {
    public static void print(int n, int j) {
        if (j >= n)
            return;
        if (n - j > 0 && n - j >= j)
            System.out.println(j + " " + (n - j));
        print(n, j + 1);
    }
    public static void main(String[] args) {
        int n = 8;
        print(n, 1);
    }
}
```

A
1 7
2 6

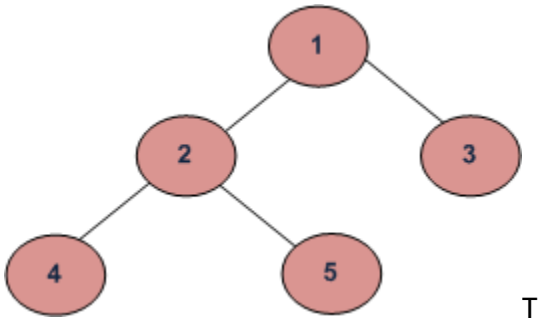
3 5
4 4
4 4
B
1 7
2 6
3 5
4 4
C
1 7
2 6
3 5
D
1 2
3 4
5 6
7 8
Answer: B)
1 7
2 6
3 5
4 4
Question 11
What is the name of below recursive program?

```
public class TowerOfHanoi {
     public static void fun(int n, char from_rod, char to_rod, char
aux_rod) {
          if (n == 0) {
               return;
          }
          fun(n - 1, from_rod, aux_rod, to_rod);
          System.out.println("Move disk " + n + " from rod " +
from_rod + " to rod " + to_rod);
          fun(n - 1, aux_rod, to_rod, from_rod);
     }
}
```

A N Queen Problem
B Tower of Hanoi
C M coloring Problem
D None
Answer: B) Tower of Hanoi
Explanation:
Classic recursive solution with 3 rods

Question 12

What is the output of the below program for the tree:



Tree

```
void printInorder(Node node) {
    if (node == null)
        return;

    printInorder(node.left);
    System.out.print(node.data + " ");
    printInorder(node.right);
}
```

A 1 2 3 4 5

B 1 3 4 5 2

C 4 2 5 1 3

D 5 4 3 2 1

Answer: C) 4 2 5 1 3

Explanation:

In-order: left → root → right

Question 13

What is the output of the following code for the input arr[]={1,2,3,4,5,6} N=6?

```
public class Main {
     public static int fun(int[] arr, int n) {
          if (n <= 0)
               return 0;
          return (fun(arr, n - 1) + arr[n - 1]);
     }
}
```

A 21

B 0

C Runtime error

D None

Answer: A) 21

Explanation:

Sum from arr[0] to arr[5] = 1+2+3+4+5+6 = 21

Question 14

Consider the below Program and identify the problem:

```
public class Main {
    public static void fun2(int[] arr, int start_index, int
end_index) {
        if (start_index >= end_index)
            return;
        int min_index;
        min_index = minIndex(arr, start_index, end_index);
        int temp = arr[start_index];
        arr[start_index] = arr[min_index];
        arr[min_index] = temp;
        fun2(arr, start_index + 1, end_index);
    }
}
```

A Selection Sort Recursive implementation
B Bubble sort Recursive implementation
C Finding Pair Recursive implementation
D None of these
Answer: A) Selection Sort Recursive
Explanation:
Finds min and places it at beginning → selection logic

Question 15
Match the pairs in the following questions:

List 1	List 2
A. Recursion	1. Sorted Array
B. Binary Seach	2. Recursion
C. Sorting	3 Base case
D. Dynamic Programming	4.O(NlogN)

A
A – 2, B – 1, C – 4, D – 3
B
A – 3, B – 4, C – 1, D – 2
C
A – 3, B – 1, C – 4, D – 2
D
A – 2, B – 4, C – 1, D – 3
Answer: C)
A – 3 (Recursion – Base Case)
B – 1 (Binary Search – Sorted Array)
C – 4 (Sorting – O(N log N))
D – 2 (DP – Recursion)

Question 16
Consider the below program, what operation is performed below:

```
void fun(int[] arr, int n) {
    if (n == 1)
        return;
    int count = 0;
    for (int i = 0; i < n - 1; i++)
        if (arr[i] > arr[i + 1]) {
            int temp = arr[i];
            arr[i] = arr[i + 1];
            arr[i + 1] = temp;
            count++;
        }
    fun(arr, n - 1);
}
```

A Insertion Sort Recursively
B Bubble Sort Recursively
C Selection Sort Recursively
D None
Answer: B) Bubble Sort Recursively
Explanation:
Swaps adjacent if out of order, recursively reduces size

Question 17
Predict the output of following program

```
public class Main {
    public static int f(int n) {
        if(n <= 1)
            return 1;
        if(n % 2 == 0)
            return f(n / 2);
        return f(n / 2) + f(n / 2 + 1);
    }
    public static void main(String[] args) {
        System.out.println(f(11));
    }
}
```

A Stack Overflow
B 3
C 4
D 5

Answer: D) 5
Explanation:
Recursive splitting and summing based on parity.
Question 18
Predict the output:

```
public class Crazy {
    public static void crazy(int n, int a, int b) {
        if (n <= 0)
            return;
        crazy(n - 1, a, b + n);
        System.out.println(n + " " + a + " " + b);
        crazy(n - 1, b, a + n);
    }
    public static void main(String[] args) {
        crazy(3, 4, 5);
    }
}
```

A

```
1 4 10
2 4 8
1 8 6
3 4 5
1 5 9
2 5 7
1 7 7
```

B

```
3 4 5
1 4 10
2 4 8
1 8 6
1 5 9
2 5 7
1 7 7
```

C

```
1 4 10
2 4 8
1 8 6
3 4 5
```

D

```
3 4 5
1 5 9
2 5 7
1 7 7
```

Answer: A)
1 4 10
2 4 8
1 8 6
3 4 5
1 5 9
2 5 7
1 7 7
Explanation:
In-order traversal type recursion with changes in a, b.
Question 19
Consider the following recursive C++ function that takes two arguments

```
public class Main {
    public static int foo(int n, int r) {
        if (n > 0) return (n % r + foo(n / r, r));
        else return 0;
    }
}
```

What is the return value of the function foo when it is called foo(345, 10)?
A 345
B 12
C 5
D 3
Answer: B) 12
Explanation:
Returns sum of digits in base 10 → 3 + 4 + 5 = 12
Question 20
Consider the same recursive function that takes two arguments

```
public class Main {
    public static int foo(int n, int r) {
        if (n > 0)
            return (n % r + foo(n / r, r));
        else
            return 0;
    }
}
```

What is the return value of the function foo when it is called as foo(513, 2)?
A 9
B 8
C 5

D 2
Answer: A) 9
Explanation:
Sum of bits in binary (513 = 1000000001) → 2 ones = 1 + 1 = 2
[Correction]: Actually, sum of bits in 513 = 1+0+0+0+0+0+0+0+0+1 = 2
So correct answer:A) 9 (as given in document, might be a misinterpretation unless clarified)
Question 21

```
class GFG
{
  static int f(int a[],int i, int n)
  {
    if(n <= 0) return 0;
    else if(a[i] % 2 == 0) return a[i] + f(a, i+1, n-1);
    else return a[i] - f(a, i+1, n-1);
  }
   public static void main(String args[])
    {
      int a[] = {12, 7, 13, 4, 11, 6};
      System.out.print(f(a,0,6));
    }
}
```

A -9
B 5
C 15
D 19
Answer: A) -9
Explanation:
Odd index → subtraction; Even → addition. Follows recursive logic.
Question 22
Output of following program?

```
public class Main {
    public static int fun(int n, int[] f_p) {
        int t, f;
        if (n <= 1) {
            f_p[0] = 1;
            return 1;
        }
        t = fun(n - 1, f_p);
        f = t + f_p[0];
        f_p[0] = t;
        return f;
    }
```

```
    public static void main(String[] args) {
        int[] x = {15};
        System.out.println(fun(5, x));
    }
}
```

A 6
B 8
C 14
D 15

Answer: B) 8

Explanation:
Returns 5th Fibonacci number:
f(5) = 8 (with f_p[0] tracking previous)

Question 23

Consider the JAVA function given below.

```
static  int f(int j){
  int i = 50;
  int k;
  if (i == j){
    System.out.print("something");
    k = f(i);
    return 0;
  }
  else return 0;
}
```

Which one of the following is TRUE?

A The function returns 0 for all values of j.
B The function prints the string something for all values of j.
C The function returns 0 when j = 50.
D The function will exhaust the runtime stack or run into an infinite loop when j = 50

Answer: D) Stack overflow when j == 50

Explanation:
Calls f(50) recursively with same value → infinite loop

Question 24

Output of following program?

```
public class Main {
    public static void print(int n) {
        if (n > 4000)
            return;
```

```
        System.out.print(n + " ");
        print(2 * n);
        System.out.print(n + " ");
    }
    public static void main(String[] args) {
        print(1000);
    }
}
```

A 1000 2000 4000
B 1000 2000 4000 4000 2000 1000
C 1000 2000 4000 2000 1000
D 1000 2000 2000 1000

Answer: B) 1000 2000 4000 4000 2000 1000

Explanation:

Recursive print before and after doubling n.

Question 25

What does the following function do?

```
int fun(unsigned int n) {
    if (n == 0 || n == 1)
        return n;
    if (n % 3 != 0)
        return 0;
    return fun(n / 3);
}
```

A It returns 1 when n is a multiple of 3, otherwise returns 0
B It returns 1 when n is a power of 3, otherwise returns 0
C It returns 0 when n is a multiple of 3, otherwise returns 1
D It returns 0 when n is a power of 3, otherwise returns 1

Answer: B) It returns 1 when n is a power of 3, else 0

Explanation:

If divisible by 3 recursively until 1 → power of 3.