1. What is the time complexity of the Fibonacci function?
A) O(n)
B) O(2^n)
C) O(n^2)
D) O(log n)
Answer: B) O(2^n)
Explanation: Each call generates two more calls, leading to exponential growth in calls.
2. What happens when a recursive function lacks a base case?
A) Runs indefinitely
B) Throws an exception
C) Runs but produces no output
D) Compiles but does nothing
Answer: A) Runs indefinitely
Explanation: Without a base case, recursion continues forever, causing a StackOverflowError.
3. Which of the following problems is best solved using recursion?
A) Finding the largest element in an array
B) Checking if a number is prime
C) Tower of Hanoi
D) Sorting an array
Answer: C) Tower of Hanoi
Explanation: Tower of Hanoi is a classic problem that is naturally expressed using recursion.
4. What is the space complexity of a recursive function that does not use extra variables?
A) O(1)
B) O(n)
C) O(log n)
D) O(n^2)
Answer: B) O(n)
Explanation: One stack frame is used for each recursive call, leading to linear space usage.
5. Which of the following sorting algorithms is based on recursion?
A) Bubble Sort
B) Quick Sort
C) Selection Sort
D) Insertion Sort
Answer: B) Quick Sort
Explanation: Quick Sort uses recursion to sort the partitions of an array.
6. What is recursion in Java?
a) A function calling another function
b) A function calling itself
c) A function that runs infinitely
d) A function with multiple parameters
Answer: b) A function calling itself
Explanation: Recursion occurs when a method calls itself directly or indirectly.

7. What is the base case in recursion?
a) The case when the function calls itself
b) The condition that stops the recursion
c) The middle element in recursion
d) The first function call
Answer: b) The condition that stops the recursion
Explanation: It prevents infinite recursion by providing a stopping point.
8. What happens if a recursive function has no base case?
a) It runs once and stops
b) It runs indefinitely (StackOverflowError)
c) It runs in a loop
d) It gives a compilation error
Answer: b) It runs indefinitely (StackOverflowError).
Explanation: Recursive functions without base cases cause infinite function calls.
9. What data structure is used for function calls in recursion?
a) Queue
b) Stack
c) Heap
d) Linked List
Answer: b) Stack.
Explanation: Java uses the call stack to manage recursive method calls.

10. What is the primary advantage of recursion?
a) It makes code easier to understand
b) It improves execution speed
c) It reduces memory usage
d) It always avoids loops
Answer: a) It makes code easier to understand.
Explanation: Recursion simplifies problems like tree traversal, factorial, etc.

11. What is the time complexity of computing the nth Fibonacci number using recursion?
a) $O(n)$
b) $O(\log n)$
c) $O(2^n)$
d) $O(n^2)$
Answer: c) $O(2^n)$.
Explanation: Each Fibonacci number is computed by recursively calculating two previous numbers.
12. Which of the following problems is best solved using recursion?
a) Sorting an array
b) Tower of Hanoi
c) Printing elements of an array

d) Multiplication of two numbers
Answer: b) Tower of Hanoi
Explanation: Each move depends on solving smaller subproblems recursively.
13. What is tail recursion?
a) A recursive call made at the beginning of the function
b) A recursive call made at the end of the function
c) A recursive call inside a loop
d) A recursive function with multiple parameters
Answer: b) A recursive call made at the end of the function.
Explanation: No operation is pending after the recursive call, allowing optimization.
14. What is the worst-case time complexity of QuickSort using recursion?
a) O(n log n)
b) $O(n^2)$
c) O(n)
d) O(log n)
Answer: b) $O(n^2)$.
Explanation: Happens when partitions are completely unbalanced (e.g., sorted input).
15. How can recursion be optimized?
a) Using loops
b) Using memoization
c) Using switch-case
d) Increasing function calls
Answer: b) Using memorization
Explanation: Caches results of subproblems to avoid redundant calls.
16. Which of the following best describes indirect recursion?
a) A function calling itself
b) Two or more functions calling each other in a cycle
c) A function that never stops
d) A function with multiple parameters
Answer: b) Two or more functions calling each other in a cycle
Explanation: For example, A calls B, and B calls A again.
17. What is Recursion in Java?
a) Recursion is a class
b) Recursion is a process of defining a method that calls other methods repeatedly
c) Recursion is a process of defining a method that calls itself repeatedly
d) Recursion is a process of defining a method that calls other methods which in turn call again this method
Answer: c
Explanation: Recursion is the process of defining something in terms of itself. It allows us to define a method that calls itself.
18. Which of these data types is used by operating system to manage the Recursion in Java?
a) Array
b) Stack

c) Queue
d) Tree
Answer: b
Explanation: Recursions are always managed by using stack.
19. Which of these will happen if recursive method does not have a base case?
a) An infinite loop occurs
b) System stops the program after some time
c) After 1000000 calls it will be automatically stopped
d) None of the mentioned
Answer: a
Explanation: If a recursive method does not have a base case then an infinite loop occurs which results in Stack Overflow.
20. Which of these is not a correct statement?
a) A recursive method must have a base case
b) Recursion always uses stack
c) Recursive methods are faster than programmer's written loop statement that calls the function iteratively
d) Recursion is managed by Java Runtime environment
Answer: c
Explanation: Recursion has always an overhead associated with a function call i.e., it has to maintain the function in the stack, manage space for variables, temp memories, and return pointers. Every recursive call will be in its own stack frame that can lead to stack overflow. An iterative method will be faster in general as it doesn't have stackframe overheads. Recursion is managed by Java Runtime only which in turn can utilize the platform APIs.
21. Which of these packages contains the error StackOverflowError in Java?
a) java.lang
b) java.util
c) java.io
d) java.system
Answer: a
Explanation: StackOverflowError is a subclass of VirtualMachineError located in the java.lang package, which contains fundamental classes and errors used by the Java runtime.
22. Recursion is a method in which the solution of a problem depends on ____________
a) Larger instances of different problems
b) Larger instances of the same problem
c) Smaller instances of the same problem
d) Smaller instances of different problems

Answer: c
Explanation: In recursion, the solution of a problem depends on the solution of smaller instances of the same problem.
23. Which of the following problems can't be solved using recursion?
a) Factorial of a number

b) Nth fibonacci number
c) Length of a string
d) Problems without base case

Answer: d
Explanation: Problems without base case leads to infinite recursion call. In general, we will assume a base case to avoid infinite recursion call. Problems like finding Factorial of a number, Nth Fibonacci number and Length of a string can be solved using recursion.
24. Recursion is similar to which of the following?
a) Switch Case
b) Loop
c) If-else
d) if elif else

Answer: b
Explanation: Recursion is similar to a loop.
25. In recursion, the condition for which the function will stop calling itself is ____________
a) Best case
b) Worst case
c) Base case
d) There is no such condition

Answer: c
Explanation: For recursion to end at some point, there always has to be a condition for which the function will not call itself. This condition is known as base case.
26. What will happen when the below code snippet is executed?
```
void my_recursive_function()
{
  my_recursive_function();
}
int main()
{
  my_recursive_function();
  return 0;
}
```
a) The code will be executed successfully and no output will be generated
b) The code will be executed successfully and random output will be generated
c) The code will show a compile time error
d) The code will run for some time and stop when the stack overflows

Answer: d
Explanation: Every function call is stored in the stack memory. In this case, there is no terminating condition(base case). So, my_recursive_function() will be called continuously till

the stack overflows and there is no more space to store the function calls. At this point of time, the program will stop abruptly.

27. What is the output of the following code?

```
void my_recursive_function(int n)
{
   if(n == 0)
   return;
   printf("%d ",n);
   my_recursive_function(n-1);
}
int main()
{
   my_recursive_function(10);
   return 0;
}
```

a) 10
b) 1
c) 10 9 8 … 1 0
d) 10 9 8 … 1

Answer: d
Explanation: The program prints the numbers from 10 to 1.

28. What is the base case for the following code?

```
void my_recursive_function(int n)
{
    if(n == 0)
    return;
    printf("%d ",n);
    my_recursive_function(n-1);
}
int main()
{
    my_recursive_function(10);
    return 0;
}
```

a) return
b) printf(“%d “, n)
c) if(n == 0)
d) my_recursive_function(n-1)

Answer: c
Explanation: For the base case, the recursive function is not called. So, “if(n == 0)” is the base case.

29. How many times is the recursive function called, when the following code is executed?

```
void my_recursive_function(int n)
{
    if(n == 0)
    return;
    printf("%d ",n);
    my_recursive_function(n-1);
}
int main()
{
    my_recursive_function(10);
    return 0;
}
```

a) 9
b) 10
c) 11
d) 12

Answer: c
Explanation: The recursive function is called 11 times.

30. What does the following recursive code do?

```
void my_recursive_function(int n)
{
    if(n == 0)
    return;
    my_recursive_function(n-1);
    printf("%d ",n);
}
int main()
{
    my_recursive_function(10);
    return 0;
}
```

a) Prints the numbers from 10 to 1
b) Prints the numbers from 10 to 0
c) Prints the numbers from 1 to 10
d) Prints the numbers from 0 to 10

Answer: c
Explanation: The above code prints the numbers from 1 to 10.

31. Which of the following statements is true?
a) Recursion is always better than iteration
b) Recursion uses more memory compared to iteration

c) Recursion uses less memory compared to iteration
d) Iteration is always better and simpler than recursion

Answer: b
Explanation: Recursion uses more memory compared to iteration because every time the recursive function is called, the function call is stored in stack.
32. What will be the output of the following code?

```
int cnt=0;
void my_recursive_function(int n)
{
    if(n == 0)
    return;
    cnt++;
    my_recursive_function(n/10);
}
int main()
{
    my_recursive_function(123456789);
    printf("%d",cnt);
    return 0;
}
```

a) 123456789
b) 10
c) 0
d) 9

Answer: d
Explanation: The program prints the number of digits in the number 123456789, which is 9.
33. What will be the output of the following code?

```
void my_recursive_function(int n)
{
   if(n == 0)
   {
       printf("False");
            return;
   }
   if(n == 1)
   {
       printf("True");
       return;
   }
   if(n%2==0)
   my_recursive_function(n/2);
```

```
    else
    {
        printf("False");
        return;
    }

}
int main()
{
    my_recursive_function(100);
    return 0;
}
```

a) True
b) False

Answer: b
Explanation: The function checks if a number is a power of 2. Since 100 is not a power of 2, it prints false.
34. What is the output of the following code?

```
int cnt = 0;
void my_recursive_function(char *s, int i)
{
    if(s[i] == '\0')
      return;
    if(s[i] == 'a' || s[i] == 'e' || s[i] == 'i' || s[i] == 'o' || s[i] == 'u')
    cnt++;
    my_recursive_function(s,i+1);
}
int main()
{
    my_recursive_function("thisisrecursion",0);
    printf("%d",cnt);
    return 0;
}
```

a) 6
b) 9
c) 5
d) 10

Answer: a
Explanation: The function counts the number of vowels in a string. In this case the number is vowels is 6.
35. What is the output of the following code?

```
void my_recursive_function(int *arr, int val, int idx, int len)
{
   if(idx == len)
   {
      printf("-1");
      return ;
   }
   if(arr[idx] == val)
   {
      printf("%d",idx);
      return;
   }
   my_recursive_function(arr,val,idx+1,len);
}
int main()
{
    int array[10] = {7, 6, 4, 3, 2, 1, 9, 5, 0, 8};
    int value = 2;
    int len = 10;
    my_recursive_function(array, value, 0, len);
    return 0;
}
```

a) 3
b) 4
c) 5
d) 6

Answer: b
Explanation: The program searches for a value in the given array and prints the index at which the value is found. In this case, the program searches for value = 2. Since, the index of 2 is 4(0 based indexing), the program prints 4.

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

```
public class Test {
   public static void printArray(int[] arr, int index) {
      if (index == arr.length) return;
      System.out.print(arr[index] + " ");
      printArray(arr, index + 1);
   }

   public static void main(String[] args) {
      int[] arr = {1, 2, 3};
```

```
        printArray(arr, 0);
    }
}
```
a) 3 2 1
b) 1 2 3
c) 1 3 2
d) Compilation error
b) 1 2 3
Explanation: The function prints each element recursively from index 0 to the end.

---

37. What is the base condition in the following recursive method?
```
public static void printArray(int[] arr, int index) {
    if (index == arr.length) return;
    System.out.print(arr[index] + " ");
    printArray(arr, index + 1);
}
```
a) index == 0
b) index == arr.length
c) index < arr.length
d) index != arr.length
Answer: b) index == arr.length
Explanation: When index equals the length, recursion stops.

---

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

```
public static void reversePrint(int[] arr, int index) {
    if (index < 0) return;
    System.out.print(arr[index] + " ");
    reversePrint(arr, index - 1);
}

public static void main(String[] args) {
    int[] arr = {4, 5, 6};
    reversePrint(arr, arr.length - 1);
}
```
a) 4 5 6
b) 6 5 4
c) 6 4 5
d) Compilation error
Answer: b) 6 5 4
Explanation: Starts from the end and prints in reverse using recursion.

---

39. What is printed by the following code?

```
public static void sum(int[] arr, int index, int total) {
    if (index == arr.length) {
        System.out.println(total);
        return;
    }
    sum(arr, index + 1, total + arr[index]);
}

public static void main(String[] args) {
    int[] arr = {1, 2, 3};
    sum(arr, 0, 0);
}
```

a) 0
b) 3
c) 6
d) 1 2 3
Answer: c) 6
Explanation: Recursively accumulates the sum of array elements.

---

40. Which of the following correctly finds the maximum element in an array using recursion?

a)
```
if (index == arr.length - 1) return arr[index];
return Math.max(arr[index], findMax(arr, index + 1));
```
b)
```
return arr[index];
```
c)
```
return arr[arr.length - 1];
```
d)
```
return Math.min(arr[index], findMax(arr, index + 1));
```
Answer: a)
Explanation: Uses Math.max recursively to find the largest element.