Q1. [Definition] What is recursion in programming?

A. A loop that calls itself

B. A function that calls itself

C. A class that creates an object

D. A variable that refers itself

Answer: B

Explanation: Recursion occurs when a function calls itself directly or indirectly.

Q2. [Definition] What is a base case in recursion?

A. The point where the function begins

B. A condition to start recursion

C. A condition where recursion stops

D. A return value of zero

Answer: C

Explanation: The base case is the condition that stops further recursive calls.

Q3. [Definition] Which of the following problems is best solved using recursion?

A. Sorting an array

B. Searching in a loop

C. Traversing tree structures

D. Iterating an array

Answer: C

Explanation: Recursive solutions are ideal for hierarchical data like trees.

Q4. [Definition] What is the main drawback of recursion?

A. Takes more time

B. Code becomes unreadable

C. Higher memory usage due to call stack

D. Cannot handle loops

Answer: C

Explanation: Recursive calls are stored in the call stack and can cause stack overflow.

Q5. [Definition] What happens if the base case is missing in a recursive function?

A. Function returns 0

B. Infinite recursion occurs

C. Compilation error

D. Function is skipped

Answer: B

Explanation: Without a base case, recursion never stops, leading to stack overflow.

Q6. [Moderate] What is the base case for a recursive factorial function?

A. n == 0

B. n == 2

C. n == 10

D. n == -1

Answer: A

Explanation: The base case is typically factorial(0) = 1.

Q7. [Moderate] What does the following code return for factorial(3)?

```
int factorial(int n) {
   if (n == 0) return 1;
   return n * factorial(n - 1);
}
```

A. 3

B. 6

C. 9

D. 2

Answer: B

Explanation: 3 * 2 * 1 = 6.

Q8. [Moderate] Which statement is true for recursive factorial?

A. It is faster than iterative

B. Stack is not used

C. It calls itself with n+1

D. It multiplies the current value with factorial(n-1)

Answer: D

Explanation: Recursive factorial follows n * factorial(n - 1).

Q9. [Moderate] What is the time complexity of the recursive factorial function?

A. O(1)

B. O(n^2)

C. O(n)

D. O(log n)

Answer: C

Explanation: One recursive call per number from n down to 1.

Q10. [Moderate] What issue will the following function have?

```
int factorial(int n) {
   return n * factorial(n - 1);
}
```

A. Compiles but returns 0

B. Causes stack overflow

C. Works for negative numbers

D. Returns infinite value

Answer: B

Explanation: Without a base case, recursion never stops.

Q11. [Moderate] What is the base case for a recursive Fibonacci function?

A. fib(0) = 1, fib(1) = 2

B. fib(0) = 0, fib(1) = 1

C. fib(1) = 0, fib(2) = 1

D. fib(1) = 1, fib(2) = 2

Answer: B

Explanation: The Fibonacci sequence starts with 0 and 1.

Q12. [Moderate] What is the output of fibonacci(4) in this code?

```
int fibonacci(int n) {
   if (n <= 1) return n;
   return fibonacci(n - 1) + fibonacci(n - 2);
}
```

A. 3

B. 5

C. 2

D. 4

Answer: A

Explanation: Sequence: 0, 1, 1, 2, 3 → fib(4) = 3

Q13. [Moderate] What is the time complexity of naive recursive Fibonacci?

A. O(n)

B. $O(n^2)$

C. $O(2^n)$

D. O(log n)

Answer: C

Explanation: Each call makes two more calls, resulting in exponential time.

Q14. [Moderate] Why is memoization helpful in recursive Fibonacci?

A. Reduces base case

B. Avoids recalculating subproblems

C. Increases stack calls

D. Multiplies values

Answer: B

Explanation: Memoization caches previous results to avoid repeated work.

Q15. [Moderate] What will happen if you compute fibonacci(50) with naive recursion?

A. Returns immediately

B. Gives correct result in 50ms

C. Stack overflow or very slow

D. Infinite loop

Answer: C

Explanation: Naive recursion for large inputs is inefficient and can cause stack overflow.

Q1. [Definition] What is the total number of subsets (power set) of a set with n elements?

A. n

B. n!

C. 2^n

D. n^2

Answer: C

Explanation: The power set of a set with n elements contains 2^n subsets including the empty set.

Q2. [Definition] What is the number of permutations of n distinct elements?

A. 2^n

B. n^n

C. n!

D. n^2

Answer: C

Explanation: There are n! ways to arrange n distinct items.

Q3. [Definition] Which of the following problems is best solved using backtracking?

A. Find maximum subarray

B. Generate power set

C. Linear search

D. Binary search

Answer: B

Explanation: Generating subsets uses recursion and backtracking to explore all combinations.

Q4. [Definition] Which algorithm is commonly used for generating all permutations of a string or array?

A. Kadane's Algorithm

B. Floyd's Algorithm

C. Backtracking

D. Sliding Window

Answer: C

Explanation: Backtracking is used to build all permutations by exploring each possible decision.

Q5. [Definition] In permutation problems, why is a visited array often used?

A. To count elements

B. To avoid duplicate combinations

C. To store all permutations

D. To maintain order

Answer: B

Explanation: A visited array tracks used elements to prevent reusing them in a single permutation.

Q6. [Definition] Which statement is true for the power set of {a, b}?

A. It has 4 subsets

B. It has 2 subsets

C. It has 3 subsets

D. It has 5 subsets

Answer: A

Explanation: 2 elements → $2^2 = 4$ subsets → {}, {a}, {b}, {a,b}.

Q7. [Definition] What is the time complexity of generating all permutations of n elements?

A. O(n)

B. $O(n^2)$

C. $O(2^n)$

D. O(n!)

Answer: D

Explanation: There are n! permutations of n distinct elements.

Q8. [Moderate] What is the base case in this subset-generating function?

```
void generateSubsets(int index, List<Integer> current, int[] nums) {
    if (index == nums.length) {
        System.out.println(current);
        return;
    }
    // recursive logic
}
```

A. index == 0

B. current.size() == nums.length

C. index == nums.length

D. index == nums.length - 1

Answer: C

Explanation: When all elements have been considered, a complete subset is printed.

Q9. [Moderate] What does the following code print for nums = [1, 2]?

```
void generateSubsets(int index, List<Integer> current, int[] nums) {
    if (index == nums.length) {
        System.out.println(current);
        return;
    }
    generateSubsets(index + 1, current, nums);
    current.add(nums[index]);
    generateSubsets(index + 1, current, nums);
    current.remove(current.size() - 1);
}
```

A. [], [1], [2], [1,2]
B. [1,2], [1], [2], []
C. [], [2], [1], [1,2]
D. All subsets in any order

Answer: D

Explanation: The function generates all subsets using recursion + backtracking.

Q10. [Moderate] Which line is responsible for backtracking in subset generation?

```
current.add(nums[index]);
generate(index + 1, current, nums);
current.remove(current.size() - 1);
```

A. add
B. recursive call
C. remove
D. index increment

Answer: C

Explanation: Removing the last added element is the backtracking step to explore the next path.

Q11. [Moderate] In permutation generation, what must be done after adding an element to the path?

A. Mark as visited
B. Reset array
C. Increase index
D. Skip to next element

Answer: A

Explanation: The element should be marked visited to avoid reuse.

Q12. [Moderate] What does this function print for nums = [1, 2]?

```
void permute(int[] nums, boolean[] visited, List<Integer> current) {
   if (current.size() == nums.length) {
      System.out.println(current);
      return;
   }
   for (int i = 0; i < nums.length; i++) {
      if (visited[i]) continue;
      visited[i] = true;
      current.add(nums[i]);
      permute(nums, visited, current);
      current.remove(current.size() - 1);
      visited[i] = false;
   }
}
```

A. [1, 2]
B. [2, 1]
C. [1, 2], [2, 1]
D. [2, 1], [1, 2], [1, 2]
Answer: C
Explanation: All permutations of 2 elements are generated with backtracking.
Q13. [Moderate] What is the backtracking step in permutation generation?
A. Recalling the recursive method
B. Removing the element and resetting visited[i]
C. Breaking the loop
D. Changing array size
Answer: B
Explanation: Backtracking is done by removing the element and marking it as unvisited.
Q14. [Moderate] What is the output of this for nums = [1, 2, 3]?
System.out.println(Math.pow(2, nums.length));
A. 8.0
B. 6.0
C. 9.0
D. 3.0
Answer: A
Explanation: Power set = 2^n = 2^3 = 8.
Q15. [Moderate] What change is required in a permutation function to allow duplicates?
A. Use HashSet
B. Avoid visited array
C. Sort the input and skip duplicates
D. Add all elements twice
Answer: C
Explanation: Sort the input and skip duplicates at the same recursive depth to avoid duplicate permutations.
Q1. [Definition] What is the objective of the N-Queens problem?
A. Place N rooks on an NxN board
B. Place N queens such that no two attack each other
C. Find the shortest path in a maze
D. Fill a grid with numbers
Answer: B
Explanation: N-Queens problem requires placing N queens such that no two threaten each other on the board.
Q2. [Definition] Which technique is most commonly used to solve the N-Queens problem?
A. Dynamic Programming
B. Divide and Conquer
C. Backtracking

D. Greedy Algorithm

Answer: C

Explanation: N-Queens is a classic backtracking problem where partial solutions are built and backtracked upon conflict.

Q3. [Definition] What constraints must be satisfied for placing a queen on the board?

A. No queen in same row

B. No queen in same column

C. No queen in same diagonal

D. All of the above

Answer: D

Explanation: Queens can attack in row, column, and both diagonals, so all must be checked.

Q4. [Definition] In a 9x9 Sudoku board, how many cells must be filled to complete the board?

A. 64

B. 81

C. 72

D. 56

Answer: B

Explanation: A 9x9 Sudoku board has 81 cells that must be filled with valid digits.

Q5. [Definition] What condition must be true in a valid Sudoku board?

A. Every row must contain unique numbers 1–9

B. Every column must contain unique numbers 1–9

C. Each 3x3 box must contain unique numbers 1–9

D. All of the above

Answer: D

Explanation: All rows, columns, and 3x3 subgrids must have unique digits from 1 to 9.

Q6. [Definition] Which approach is used to solve Sudoku efficiently in programming?

A. Sliding window

B. Divide and conquer

C. Backtracking

D. Dynamic programming

Answer: C

Explanation: Sudoku solvers typically use backtracking with constraint checks.

Q7. [Definition] Why is backtracking suitable for constraint satisfaction problems like Sudoku?

A. It sorts the data

B. It generates random solutions

C. It tries partial solutions and abandons invalid ones

D. It uses recursion only

Answer: C

Explanation: Backtracking prunes invalid solutions early and only explores valid paths.

Q8. [Moderate] What is the time complexity of solving N-Queens using backtracking?

A. $O(n^2)$

B. $O(n!)$

C. $O(2^n)$

D. $O(n^3)$

Answer: B

Explanation: For each row, we try n columns, leading to a factorial complexity in the worst case.

Q9. [Moderate] What is the role of the isSafe function in N-Queens backtracking?

A. To place a queen

B. To update the board

C. To check if placing a queen causes conflicts

D. To print the board

Answer: C

Explanation: isSafe checks if a queen can be safely placed in a given row and column.

Q10. [Moderate] What would this check ensure in a Sudoku solver?

```
if (board[row][i] == num || board[i][col] == num ||
   board[row - row % 3 + i / 3][col - col % 3 + i % 3] == num)
```

A. num is in same diagonal

B. num is repeated in subgrid

C. num is unique in row, column, and 3x3 box

D. num is valid only in one column

Answer: C

Explanation: This single condition ensures the digit isn't already in the current row, column, or subgrid.

Q11. [Moderate] What's the backtracking step in Sudoku solving?

A. Try next cell

B. Skip empty cells

C. Remove previously filled digit when conflict arises

D. Return if grid is full

Answer: C

Explanation: When a placement leads to conflict later, the digit is removed and a new digit is tried.

Q12. [Moderate] What is the base case for the Sudoku recursive backtracking function?

A. When all rows are valid

B. When all digits are used

C. When all 81 cells are filled

D. When first row is filled

Answer: C

Explanation: The recursion ends when all 81 cells are filled with valid digits.

Q13. [Moderate] In N-Queens, what is the advantage of using a boolean array for columns and diagonals?

A. It speeds up board printing

B. It reduces space usage

C. It provides O(1) conflict check

D. It avoids stack overflow

Answer: C

Explanation: Arrays track queen placements and allow constant-time conflict checks.

Q14. [Moderate] What will happen in a Sudoku solver if a digit is placed without checking constraints?

A. The solution will be faster

B. The board will still be valid

C. Invalid solution or infinite recursion

D. Duplicates will be handled later

Answer: C

Explanation: Placing invalid digits may break constraints and cause infinite recursion or wrong solution.

Q15. [Moderate] In N-Queens, what do the values row - col and row + col represent when stored in HashSets?

A. Horizontal threats

B. Vertical threats

C. Diagonal threats

D. Row indices

Answer: C

Explanation: These expressions represent diagonals (main and anti-diagonals) in constant-time diagonal checks.

Q1. [Definition] What is the objective of the Combination Sum problem?

A. Find all unique permutations of a list

B. Find all combinations of numbers adding up to a target

C. Find maximum subarray sum

D. Sort the input list

Answer: B

Explanation: Combination Sum problems aim to find all sets of elements that sum up to a specific target.

Q2. [Definition] Which approach is commonly used to solve Combination Sum problems?

A. Divide and conquer

B. Dynamic programming

C. Backtracking

D. Greedy algorithm

Answer: C

Explanation: Backtracking allows exploring all valid combinations efficiently by pruning invalid paths.

Q3. [Definition] In Combination Sum, which statement is true if an element can be used multiple times?

A. Skip the current index after choosing

B. Use index+1 for the next call

C. Reuse the same index after choosing

D. Remove the element from the list

Answer: C

Explanation: Reusing the same index allows using the same element multiple times.

Q4. [Definition] What does the term "backtrack" mean in backtracking algorithms?

A. Jump to the last recursive call

B. Move to the middle of the array

C. Remove the last added element and explore new paths

D. Go back to base case

Answer: C

Explanation: Backtracking involves undoing the last action to explore other valid options.

Q5. [Definition] What is the base case in most backtracking problems?

A. When the input array is full

B. When the recursion depth equals 1

C. When a valid solution is formed (like target reached)

D. When index is negative

Answer: C

Explanation: The base case typically checks if a complete/valid solution has been formed.

Q6. [Definition] What is the advantage of backtracking over brute force?

A. Always finds optimal result

B. Explores only valid and promising paths

C. Uses more memory

D. Avoids recursion

Answer: B

Explanation: Backtracking avoids unnecessary paths by pruning invalid options early.

Q7. [Definition] Which of the following problems can be solved using backtracking?
A. Merge sort
B. Combination sum
C. Binary search
D. Heap sort
Answer: B
Explanation: Backtracking is a typical approach to solving combination sum problems.

🔷 Part 2: [Moderate] – Code, Logic, and Templates (Q8–Q15)
Q8. [Moderate] What does this recursive call ensure in combination sum?
java
CopyEdit
```java
backtrack(i, target - candidates[i], path);
```
A. Removes duplicates
B. Uses a smaller candidate
C. Reuses same element
D. Switches to next index
Answer: C
Explanation: Calling with the same index i allows the same candidate to be reused.

Q9. [Moderate] What condition should be checked to stop recursion early in combination sum?
A. if (target > 0) return;
B. if (target < 0) return;
C. if (i == 0) return;
D. if (path.size() > 0) return;
Answer: B
Explanation: If the target becomes negative, the current path is invalid and should be pruned.

Q10. [Moderate] In this template, which line is responsible for backtracking?
java
CopyEdit
```java
path.add(candidates[i]);
backtrack(i, target - candidates[i], path);
path.remove(path.size() - 1);
```
A. First line
B. Second line
C. Third line
D. No line

Answer: C

Explanation: Backtracking removes the last decision to try the next option.

Q11. [Moderate] In Combination Sum II (each number used once), what change is made in recursion?

A. Call with same index
B. Call with i-1
C. Call with i+1
D. Skip the current number

Answer: C

Explanation: Calling with i+1 ensures each number is used at most once.

Q12. [Moderate] How do you avoid duplicate combinations in Combination Sum II?

A. Do not sort the array
B. Use a visited array
C. Sort and skip duplicates in loop
D. Randomize the array

Answer: C

Explanation: Sorting allows checking if (i > start && nums[i] == nums[i-1]) continue to skip duplicates.

Q13. [Moderate] What does the following combination sum template achieve?

java
CopyEdit

```java
void backtrack(int start, int target, List<Integer> path) {
    if (target == 0) {
        result.add(new ArrayList<>(path));
        return;
    }
    for (int i = start; i < nums.length; i++) {
        if (nums[i] > target) break;
        path.add(nums[i]);
        backtrack(i, target - nums[i], path);
        path.remove(path.size() - 1);
    }
}
```

A. Finds subsets
B. Finds permutations
C. Finds all valid combinations summing to target
D. Finds duplicates

Answer: C

Explanation: This is a backtracking template for finding combinations that sum to a target.

Q14. [Moderate] In backtracking, why is path.removeLast() (or equivalent) required?

A. It optimizes memory

B. It reverses the array

C. It prevents the previous element from remaining in the next path

D. It triggers recursion

Answer: C

Explanation: Backtracking removes the element before trying new combinations.

Q15. [Moderate] What is the time complexity of the combination sum problem (worst case)?

A. O(n)

B. $O(n^2)$

C. $O(2^n)$

D. Exponential

Answer: D

Explanation: The total number of combinations to explore is exponential in the number of candidates.

Q1. [Definition] A man is standing in front of a set of mirrors. Each mirror reflects into another. This is similar to which concept?

A. Loop

B. Recursion

C. Hashing

D. Sorting

Answer: B

Explanation: Mirrors reflecting each other is like a function calling itself recursively.

Q2. [Definition] A chef is baking a cake, and inside the cake is another smaller cake, and another, and so on until a final base. What does this represent in recursion?

A. Iteration

B. Base case

C. Infinite loop

D. Sorting

Answer: B

Explanation: The final smallest cake represents the base case where recursion stops.

Q3. [Definition] A detective follows clue after clue, each pointing to the next. What does this sequence resemble?

A. Stack

B. Queue
C. Recursion
D. Graph traversal
Answer: C
Explanation: Each clue leads to another, like recursive function calls.

Q4. [Definition] A Russian doll contains another smaller doll, and so on until no more dolls. What does the smallest doll represent?
A. Recursive call
B. Base case
C. Loop break
D. Infinity
Answer: B
Explanation: The smallest doll is like the base case that ends the recursion chain.

Q5. [Definition] In a maze, you try one path. If it fails, you backtrack and try another. What concept is this?
A. Sorting
B. Dynamic programming
C. Backtracking
D. Binary search
Answer: C
Explanation: Backtracking tries all possible paths and reverts when a dead-end is hit.

Q6. [Definition] A librarian stacks books one over another. To find the first book, he must remove all above. Which data structure supports this?
A. Queue
B. Stack
C. Array
D. Linked list
Answer: B
Explanation: Stack follows LIFO, and recursion uses the call stack in the same way.

Q7. [Definition] A snake eats food and grows by repeating its pattern. This growing body structure is like?
A. ArrayList
B. HashMap
C. Recursion
D. Greedy algorithm
Answer: C
Explanation: The snake growing by repeating parts is like recursive pattern building.

Q8. [Definition] A magician picks cards and arranges them in every possible order. What concept is being used?

A. Fibonacci

B. Subsets

C. Permutations

D. Stack

Answer: C

Explanation: All possible orderings = permutations.

Q9. [Definition] A robot explores all rooms in a house, ensuring it visits every unique combination of open/closed doors. What concept is this?

A. Sorting

B. Permutation

C. Subsets / Power Set

D. Binary search

Answer: C

Explanation: Exploring all open/close combinations of doors represents subsets.

Q10. [Definition] A person is selecting outfits. She can reuse items, and wants combinations that total a budget. What algorithm helps her?

A. Merge Sort

B. Greedy

C. Combination Sum

D. Permutation

Answer: C

Explanation: Choosing combinations (with or without repetition) that total a value is Combination Sum.

Q11. [Definition] A chef tries adding ingredients to a recipe, one by one, and removes the last if the taste goes wrong. This cooking process resembles?

A. Greedy

B. Brute force

C. Backtracking

D. Memoization

Answer: C

Explanation: Adding/removing ingredients to test different outcomes mirrors backtracking.

Q12. [Definition] In a pyramid puzzle, every step depends on the two blocks below it. What famous recursive problem does this resemble?

A. Factorial

B. Permutations
C. Fibonacci
D. Subset
Answer: C
Explanation: Fibonacci depends on the sum of the two previous results.

Q13. [Definition] A janitor visits rooms in a hotel recursively. He marks visited and backs up when no more paths. Which logic is this?
A. DFS with recursion
B. BFS
C. Sorting
D. Queue traversal
Answer: A
Explanation: Recursive DFS explores and backtracks, like the janitor.

Q14. [Definition] A teacher gives students all possible groupings of 3 questions from a list of 5. Which structure is he using?
A. Permutation
B. Subset
C. Combination
D. Factorial
Answer: C
Explanation: Choosing unique groupings of questions is combination logic.

Q15. [Definition] A queen on a chessboard threatens rows, columns, and diagonals. Placing N queens so no one attacks another is solved using?
A. Recursion + Backtracking
B. Graph
C. Brute force
D. Sorting
Answer: A
Explanation: The N-Queens problem is solved by placing queens recursively and backtracking on conflicts.

Q16. [Definition] A painter has 3 colors and must color a board such that no two adjacent tiles are same. What approach will help?
A. Binary search
B. BFS
C. Backtracking with recursion
D. Sorting

Answer: C

Explanation: Trying every color and reverting on conflict is classic backtracking.

Q17. [Definition] A child builds a tower by stacking blocks one at a time, and undoes the last one to try a different color. What is this process?

A. Brute force

B. Dynamic programming

C. Backtracking

D. Sorting

Answer: C

Explanation: The act of placing and removing represents recursive backtracking.

Q18. [Definition] A player fills a Sudoku board by checking if each digit fits before placing. If stuck, he erases and tries another. Which method is used?

A. Recursion only

B. Greedy

C. Backtracking

D. Hashing

Answer: C

Explanation: Sudoku solvers use recursion with backtracking to fill valid digits.

Q19. [Definition] A climber ascends a staircase, taking 1 or 2 steps at a time. How many ways to reach the top? Which recursion fits here?

A. Permutation

B. Subset

C. Fibonacci recursion

D. Combination sum

Answer: C

Explanation: This problem maps to Fibonacci recursion.

Q20. [Definition] A person finds a solution by trying every path, and reverts whenever the current path is invalid. Which problem-solving strategy does this describe?

A. Greedy

B. BFS

C. Backtracking

D. Divide and conquer

Answer: C

Explanation: Trying and reverting invalid paths is the core idea of backtracking.