VIVA QUESTIONS & ANSWERS (AI & SP LAB)

WEEK 1 – PROLOG

1. Fact: Basic true statement in Prolog.
2. Rule: Conditional truth using other facts.
3. Query: A question asked to Prolog.
4. Backtracking: Trying alternative solutions.
5. Unification: Matching variables and constants.
6. Difference: Fact = truth, Rule = condition.
7. Recursion: Rule calling itself.
8. Inference Engine: Solves queries.
9. Meaning: parent(X,Y) $\rightarrow$ X is parent of Y.
10. Knowledge Base: Facts + rules.

WEEK 2 – EXPERT SYSTEM

1. Expert System: Mimics human decision.
2. Knowledge Base: Facts + rules.
3. Forward Chaining: Facts $\rightarrow$ conclusion.
4. Backward Chaining: Goal $\rightarrow$ check rules.
5. Example: Medical diagnosis.
6. Inference: Deriving new facts.
7. Goal: Final state to prove.
8. Difference: Facts = true, Rules = conditional.
9. Backward chaining use: Diagnosis.
10. Trait: Property of animals.

WEEK 3 – DFS/BFS

1. DFS: Depth-first search.
2. BFS: Breadth-first search.
3. DFS uses stack.
4. BFS uses queue.
5. BFS finds shortest path.
6. Complexity: O(V+E).
7. Adjacency list: Node neighbors.
8. DFS vs BFS: Deep vs wide.
9. DFS is recursive.

10. DFS detects cycles.

WEEK 4 – A* SEARCH

1. A*: Uses cost + heuristic.
2. Heuristic: Cost estimate.
3. f(n)=g(n)+h(n)
4. g(n)=actual cost.
5. h(n)=estimated cost.
6. Optimal because heuristic.
7. Uses priority queue.
8. A* vs Dijkstra: Heuristic added.
9. Admissible heuristic: Never overestimates.
10. Output: Shortest path.

WEEK 5 – MINIMAX

1. Minimax: Game decision algorithm.
2. Maximizing player: AI.
3. Minimizing player: Human.
4. Evaluation: Score function.
5. AI never loses: Checks all moves.
6. Game tree: All game states.
7. Terminal: Win/lose/draw.
8. Depth: Moves ahead.
9. Slow: Explores full tree.
10. AI uses O.

WEEK 6 – TWO PASS ASSEMBLER

1. Two-pass: Pass1 table, Pass2 code.
2. Pass1: Build symbol table.
3. Pass2: Machine code.
4. Symbol table: Label→address.
5. Opcode: Machine instruction code.
6. Operand: Data/address.
7. Error: Undefined symbol.
8. Labels: Jump references.
9. Input format: label opcode operand.

10. Output: machine code.

WEEK 7 – MACRO PROCESSOR

1. Macro: Block of code.
2. MNT: Macro Name Table.
3. MDT: Macro Definition Table.
4. Macro call: Using macro.
5. Pass1: Store definitions.
6. Pass2: Expand macros.
7. Use: Avoid repeated code.
8. MEND: End macro.
9. Argument: Placeholder.
10. Expansion: Replace macro.

WEEK 8 – SHELL IN C

1. Shell: Command interpreter.
2. fork(): Create child.
3. execvp(): Run command.
4. waitpid(): Wait child.
5. Tokenization: Split input.
6. argv: Arguments.
7. Infinite loop: Shell runs always.
8. Prompt: myshell$
9. exit: End program.
10. System call: OS function.

WEEK 9 – SHELL SCRIPT

1. Shell script: File of commands.
2. chmod +x: Make executable.
3. PID: Process ID.
4. Background process: sleep 10 &
5. ps: Show processes.
6. kill: Stop process.
7. cat: Read file.
8. mv: Rename/move.
9. rm: Delete file.

10. echo: Print text.

WEEK 10 – IPC

1. Pipe: Communication channel.
2. Read end: pipefd[0]
3. Write end: pipefd[1]
4. SIGUSR1: User signal.
5. kill(): Send signal.
6. pause(): Wait signal.
7. Child→parent: Notify after reading.
8. Close ends: Avoid deadlock.
9. IPC: Inter-process communication.
10. Buffer: Temporary data.

WEEK 11 – AI SYSTEM

1. Expert logic: Rule-based.
2. df: Disk usage.
3. uptime: System running time.
4. free: Memory info.
5. Cleanup: If disk>80%.
6. Reboot: If uptime>7 days.
7. Memory alert: Free <500 MB.
8. Rule: Condition + action.
9. Automation: Auto tasks.
10. Confirmation: User approval.

WEEK 12 – MINI PROJECT

1. Objective: Intelligent file manager.
2. Natural-language: Human-like commands.
3. AI logic: Keyword-based.
4. list: Show files.
5. find: Search files.
6. delete: Simulated removal.
7. os library: File ops.
8. Simulation: Safe delete.
9. Usefulness: Easy for users.

10. Improvement: Voice/ML.