9th June

Problem 1: Guddu Bhaiya and Enchanted Maze

Approach:
- Start from (0,0) and explore all right and down direction.
- Avoid blocked ('X') or out of bound cells.
- If you reach (N-1, M-1), print the full path.
- use recursion & backtracking to explore all valid paths.

pseudo-code:

```
void findpaths(char[][] maze,
int n, int m, int x, int y,
List<String> path){
if(x==n-1 && y==m-1){
S.O.p("[" + String.join("->", path) + "]");
return;
} // move right
if (isValid) findpaths;
path.remove

// MoveDown
if (isValid(Maze, n, m, x+1, y)){
path.add("(" + (x+1) + "," + y + ")");
findpaths(maze, n, m, x+1, y, path);
path.remove(path.size() - 1);
}
}
```

Dry run:

Start at (0,0)

path = ["(0,0)"]

1. from (0,0)

move right to (0,1)

→ path : [(0,0) → (0,1)]

2. from (0,1)

Right to (0,2) is x - blocked

Move down to (1,1)

3. from (1,1)

Move right to (1,2)

path → [(0,0) → (0,1) → (1,1) → (1,2)]

4. from (1,2)

. Right to (1,3) is x → blocked

. Move down to (2,2)

path :- [(0,0) → (0,1) → (1,1) → (1,2) → (2,2)]

5. from (2,2) :- Move right to (2,3)

path:

(0,0) → (0,1) → (1,1) → (1,2) → (2,2) → (2,3)

Problem 2: String Reversal Magic by sage Vishwakarma.

Approach : 1. Split the string into 2 equal halves
2. Recursively apply the same logic to each half.
3. Then swap the 2 halves and concatenate them.

pseudo-code:

```
if (S.length() == 1) return S;
int mid = S.length()/2;
String left = S.substring(0, mid);
String right = S.substring(mid);
return reverseMagic(right) + reverseMagic
    (left);
}
```

Dry run: I/P : S = "abcdefgh"
O/P : badcfehg

=> . abcd | efgh
• ab - cd - ef - gh
• ab - ba, cd → dc, ef → fe,
gh → hg
• ba + dc - badc, fe + hg - fehg
• badc + fehg → badcfehg =