9 June.

Date : __/__/____

1. Guddu Bhaiya & the enchanted Maze of Mirzapur.

Approach:

→ start at (0,0).

→ At every step, move Right or Down if:

  → inside the grid.

  → Its an open cell ('0')

→ If we reach (N-1, M-1), print the path.

→ Use a recursive function with Parameters: current position (x, y), path so far, & the maze.

Java code:

```
Public class GuddyEscape {
    static int N, M;
    static char[][] maze;
static void findPaths(int x, int y, List<String> path){
    if (x == N-1 && y == M-1) {
        path.add("(" + x + ", " + y + ")");
        System.out.println(path);
        path.remove(path.size()-1);
        return;
    }
    if(x<0 || y<0 || x>=N || y>=M || maze[x][y]=='
        return;
path.add("(" + x + ", " + y + ")");
    maze[x][y] = 'x';
findPaths(x, y+1, path);
findPaths(x+1, y, path);
    maze[x][y] = '0';
path.remove(path.size()-1); }
    }.
```

Date : __/__/____

Dry Run:

Possible Path:
Right → Down
Down → Right

1. findPaths (0, 0, [])
→ add (0,0)
→ move Right: (0, 1, [(0,0)])
→ add(0,1)
→ move right: out of board
→ move down:
→ reach destination! print [(0,0), (0,1), (1,1)]

→ move down: (1,0, [(0,0)])
→ add (1,0)
→ move right: (1, 1, [(0,0)], (1,0)])
→ reached destination! print [(0,0), (1,0), (1,1)]

Date : __ / __ / ____

2. String Reversal Magic by Sage Vishwakarma.

Approach.

→ Base case: if string length is 1 → return.
→ Recursive case:
  split string into two halves.
  Recursively process both halves.
  Swap their positions & merge back.

Java Code:

```
public class StringReverseMagic {
Static string reverseMagic (String s) {
  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);
  }.

}.
```

Date : __ / __

dry run:

→ abcdefgh
↓ ↓
abcd efgh

swap → efgh + abcd.

→ efgh
↓ ↓
ef gh swap → gh + ef.

→ abcd
↓ ↓
ab cd swap → cd + ab.

→ ef
↓ ↓
e f swap → f + e.

→ gh
↓ ↓
g h swap → h + g

→ ab
↓ ↓
a b swap → b + a.

→ cd
↓ ↓
c d swap d + c.

final: ghef + cdab ⇒ ghefcdab.