```
/*
    Time complexity: O(N * M)
    Space complexity: O(N * M)

    where N and M are the rows and columns respectively of the board
*/
public class Solution {

        int[][] a = {{-1,-1},{-1,0},{-1,1},{0,-1},{0,1},{1,-1},{1,0},{1,1}};
        String pattern = "CODINGNINJA";
        int[][] used;

        int validPoint(int x,int y,int N,int M){

                if(x >= 0 && x < N && y >= 0 && y < M)
                        return 1;
                return 0;
        }

        int DFS(String[] G,int x, int y, int index, int N, int M){

                if(index == 11)
                        return 1;

                used[x][y] = 1;
                int i, newx, newy;
                int found = 0;
                for(i = 0; i < 8; i++){

                        newx = x + a[i][0];
                        newy = y + a[i][1];

                        if(validPoint(newx,newy,N,M) == 1 && G[newx].charAt(newy) == pattern.charAt(index) && used[newx][newy] ==
0){
                                found = found | DFS(G,newx,newy,index+1,N,M);
                        }
                }

                used[x][y] = 0;

                return found;
        }

        int solve(String[] Graph , int N, int M)
        {
                int i,j,found = 0;

                used = new int[N][M];
```

```
		for(i = 0;i < N; i++){
			for( j = 0 , found = 0; j < M; j++ ){
				if(Graph[i].charAt(j)== 'C'){
					found = DFS(Graph,i,j,1,N,M);
					if(found == 1)
						break;
				}
			}
			if(found == 1)
				break;
		}

		return found;

	}


}
```