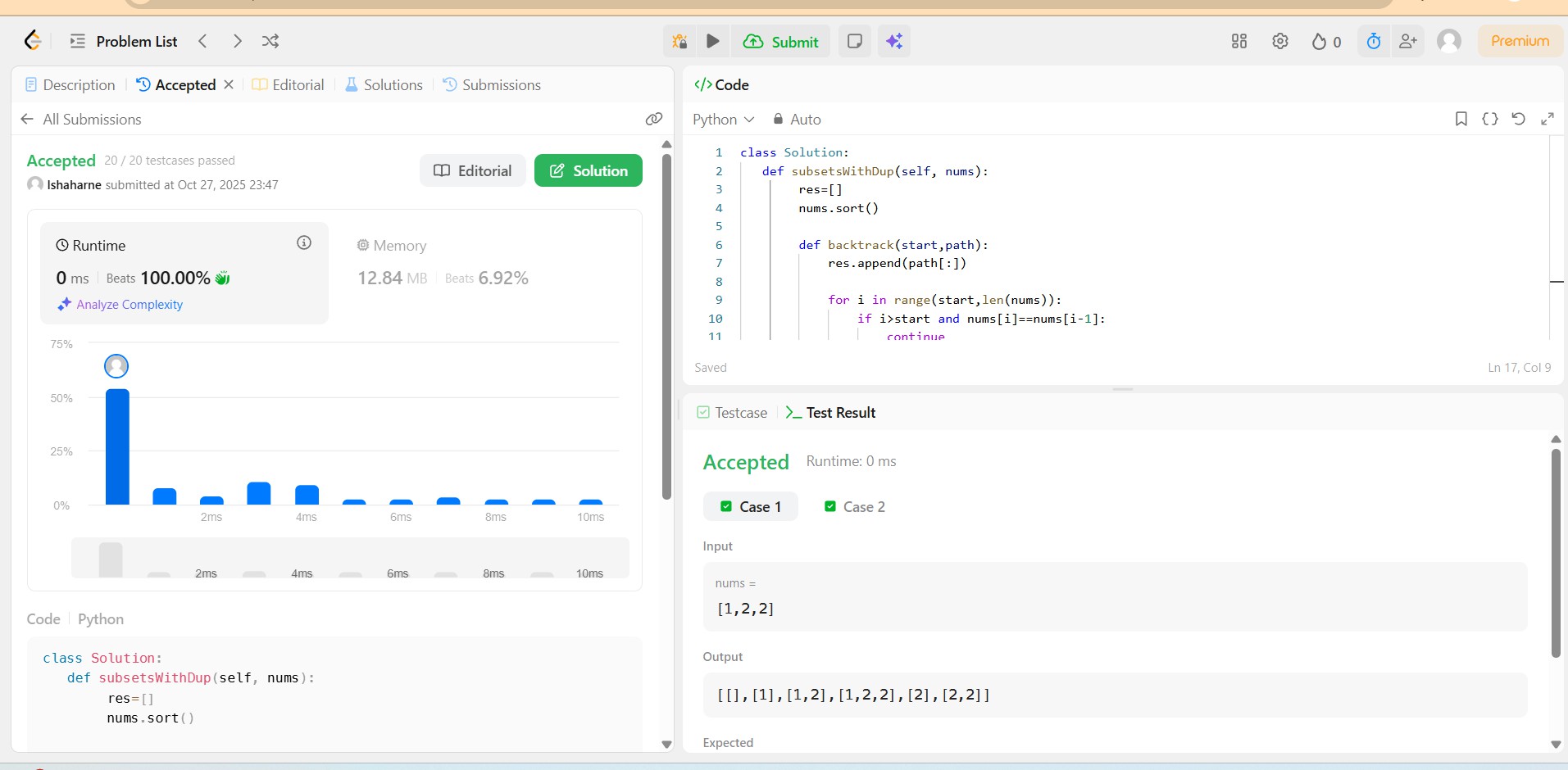
TECHNICAL SKILLS- TASK 2

Name: Neer Awasthi Class: A4 B2

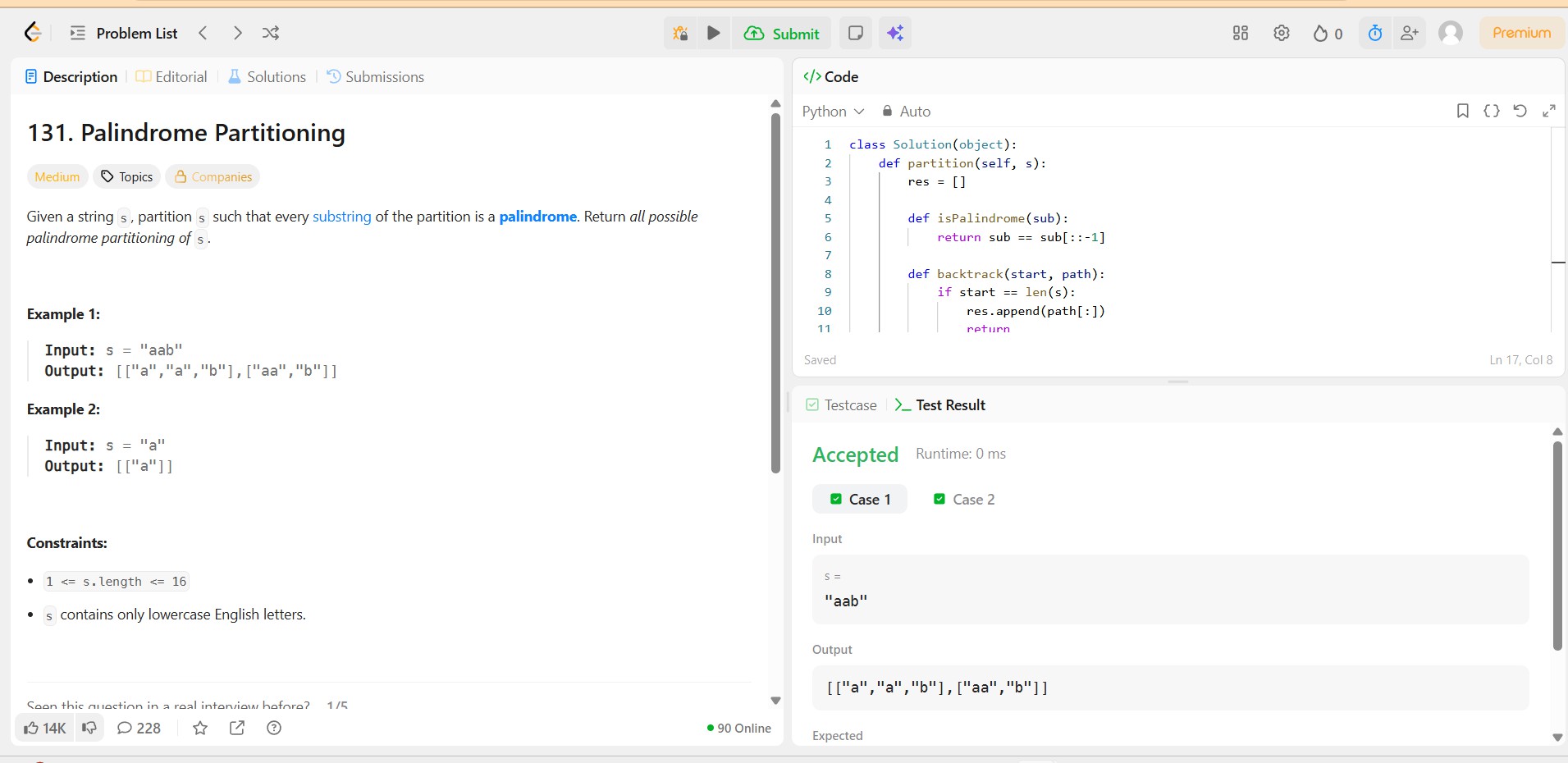
Roll No:20

Leetcode Questions

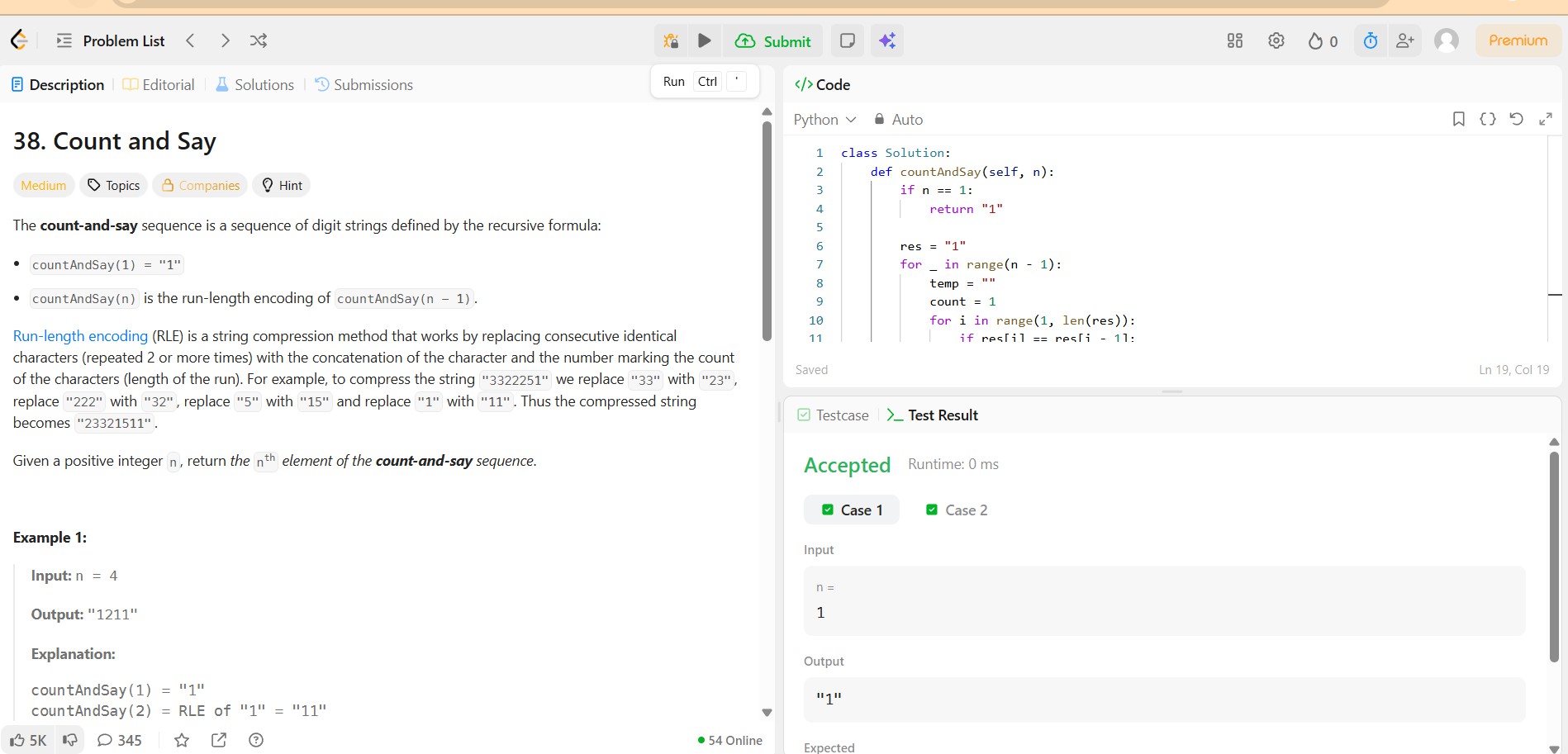
1



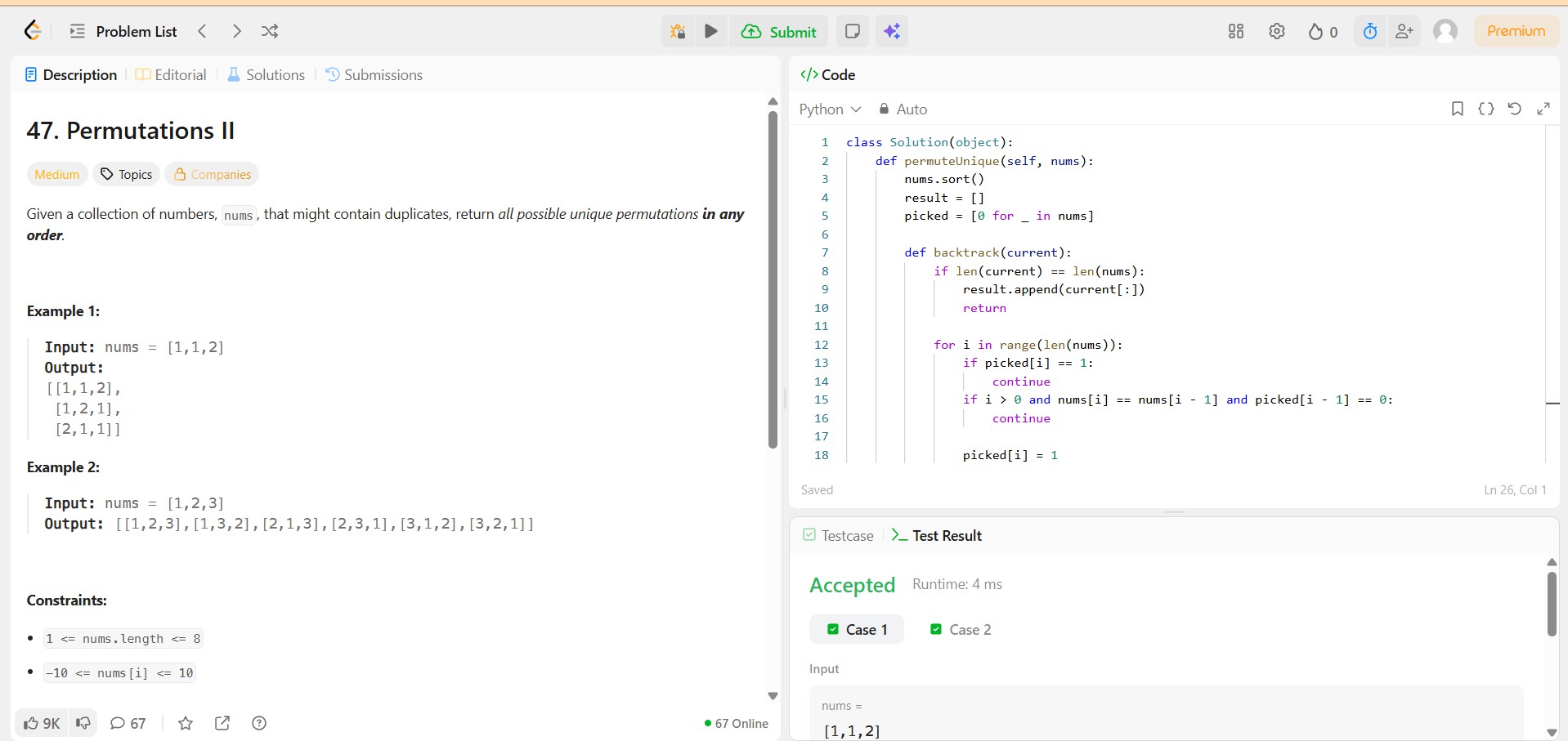
2.



3.

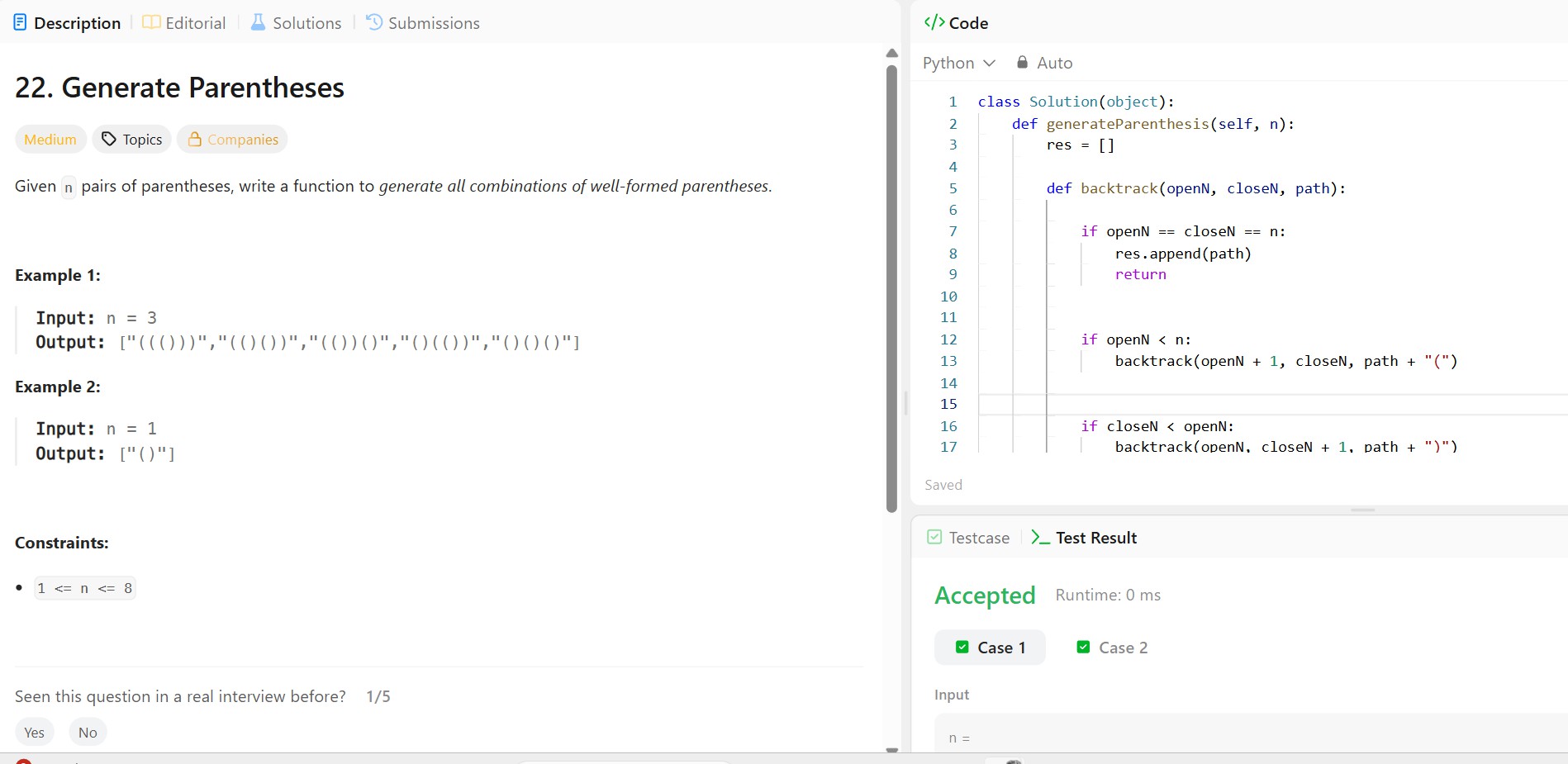


4.

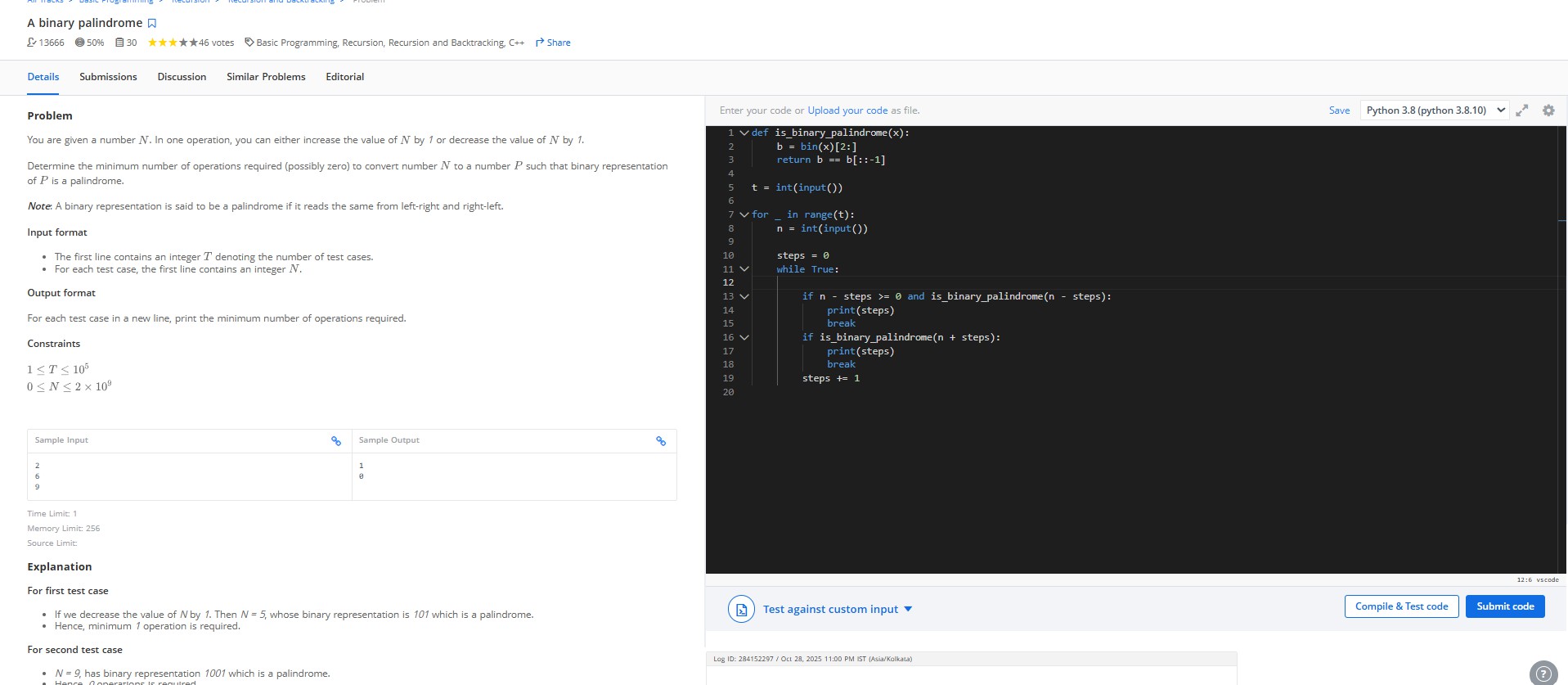


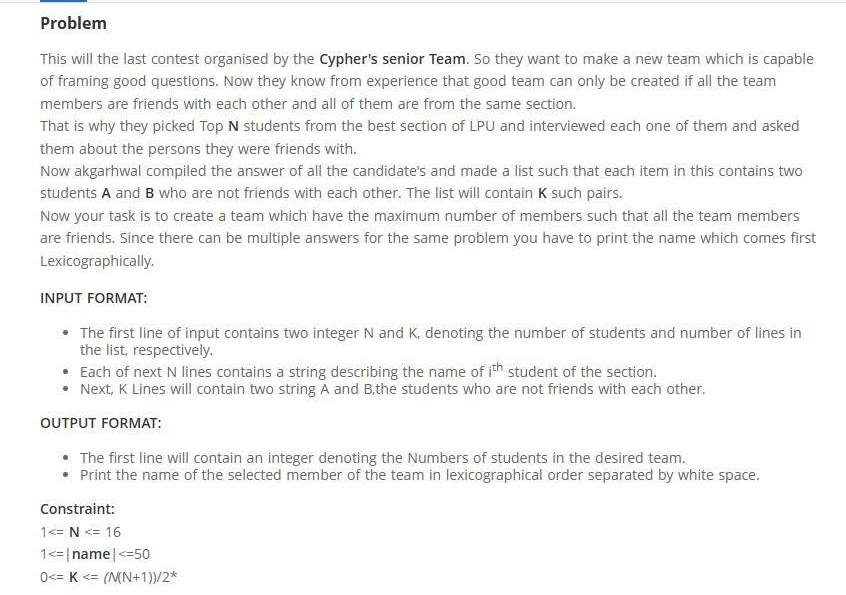
5.

5.



# Hackerearth Questions:

1. 



Code:

#include <stdio.h> #include <string.h>

#define MAXN 16

int main() {

int N, K; scanf("%d

%d", &N, &K);

char names[MAXN][51]; for (int i = 0; i < N; i++)

scanf("%s", names[i]);

int notFriend[MAXN][MAXN] = {0};

for (int i = 0; i < K; i++) { char A[51], B[51]; scanf("%s %s", A,

B); int idxA = -1, idxB = -1; for (int j = 0; j < N; j++) { if

(strcmp(names[j], A) == 0) idxA = j;

if (strcmp(names[j], B) == 0) idxB = j;

}

if (idxA != -1 && idxB != -1) { notFriend[idxA][idxB] = 1;

notFriend[idxB][idxA] = 1;

}

}

int maxSize = 0;

char bestTeam[MAXN][51]; int totalSubsets = 1 << N;

for (int mask = 1; mask < totalSubsets; mask++) { int valid = 1;

int count = 0;

char tempTeam[MAXN][51]; int idxList[MAXN];

for (int i = 0; i < N; i++) { if (mask & (1 << i)) { strcpy(tempTeam[count], names[i]);

idxList[count++] = i;

}

}

for (int i = 0; i < count && valid; i++) { for (int j = i + 1; j < count && valid; j++) {

if (notFriend[idxList[i]][idxList[j]]) valid = 0;

}

}

if (valid) { for (int i = 0; i < count - 1; i++) { for (int j = i + 1; j < count; j++) { if (strcmp(tempTeam[i], tempTeam[j]) > 0) {

char t[51]; strcpy(t, tempTeam[i]);

strcpy(tempTeam[i], tempTeam[j]); strcpy(tempTeam[j], t);

}

}

}

if (count > maxSize) {

maxSize = count; for (int i = 0; i < count; i++)

strcpy(bestTeam[i], tempTeam[i]); } else if (count == maxSize && count > 0)

{

int cmp = 0;

for (int i = 0; i < count; i++) {

cmp = strcmp(tempTeam[i], bestTeam[i]); if (cmp < 0) { for

(int j = 0; j < count; j++) strcpy(bestTeam[j], tempTeam[j]);

break;

} else if (cmp > 0) break;

}

}

}

}

printf("%d\n", maxSize); for (int i = 0; i < maxSize; i++)

{ printf("%s",

bestTeam[i]); if (i !=

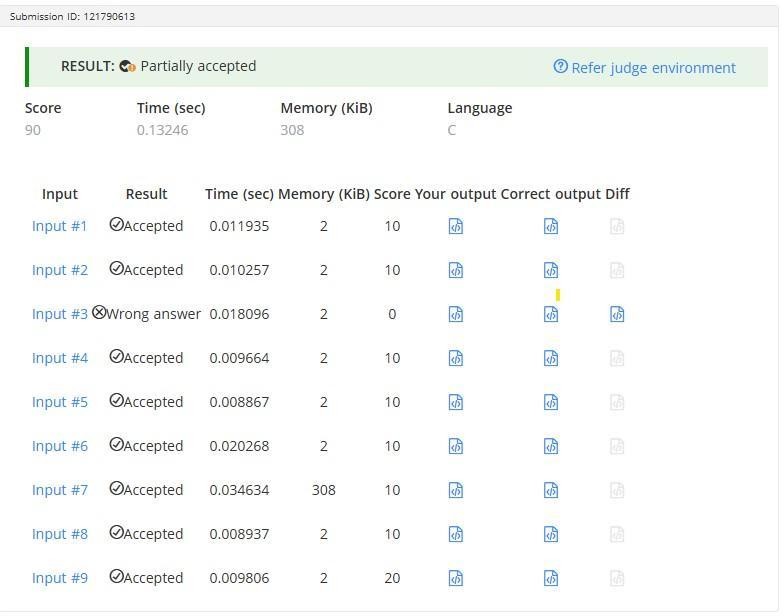
maxSize - 1) printf(" ");

}

printf("\n");

return 0;

}



3

# Code :

#include <stdio.h>

int countBought(int prices[], int n, long long wealth) { int count = 0; for (int

i = 0; i < n; i++) { if (wealth >= prices[i]) { wealth -= prices[i]; count++;

}

}

return count;

}

int main() { int n, k;

scanf("%d %d", &n, &k);

int prices[n]; long long sum = 0; for (int i

= 0; i < n; i++) {

scanf("%d", &prices[i]); sum += prices[i];

}

long long low = 0, high = sum, ans = -1;

while (low <= high) { long long mid = (low + high) / 2; int bought = countBought(prices, n, mid);

if (bought == k) { ans = mid;

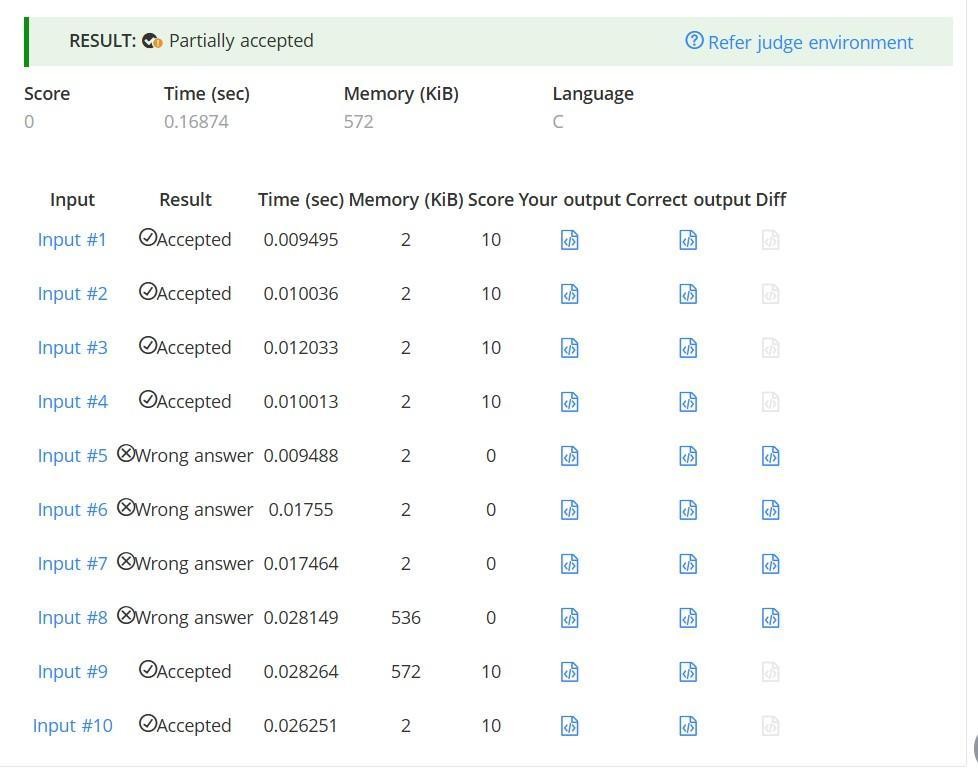
low = mid + 1; } else if (bought < k) { low = mid + 1; } else { high = mid - 1;

}

}

printf("%lld\n", ans >= 0 ? ans : 0); return 0;

}



Q.3

# Code:

#include <string.h>

#define MAX 100000

int main() { int t;

scanf("%d", &t);

while (t--) {

int n; char s[MAX];

scanf("%d", &n);

scanf("%s", s); char minSuffix[MAX]; minSuffix[n - 1] = s[n - 1]; for (int

i = n - 2; i >= 0; i--) {

if (s[i] < minSuffix[i + 1])

minSuffix[i] = s[i]; else

minSuffix[i] = minSuffix[i + 1];

}

char stack[MAX], result[MAX]; int top = -1, resIndex = 0;

for (int i = 0; i < n; i++) { stack[++top] = s[i];

while (top >= 0 && (i == n - 1 || stack[top] <= minSuffix[i + 1])) { result[resIndex++] = stack[top--];

}

}

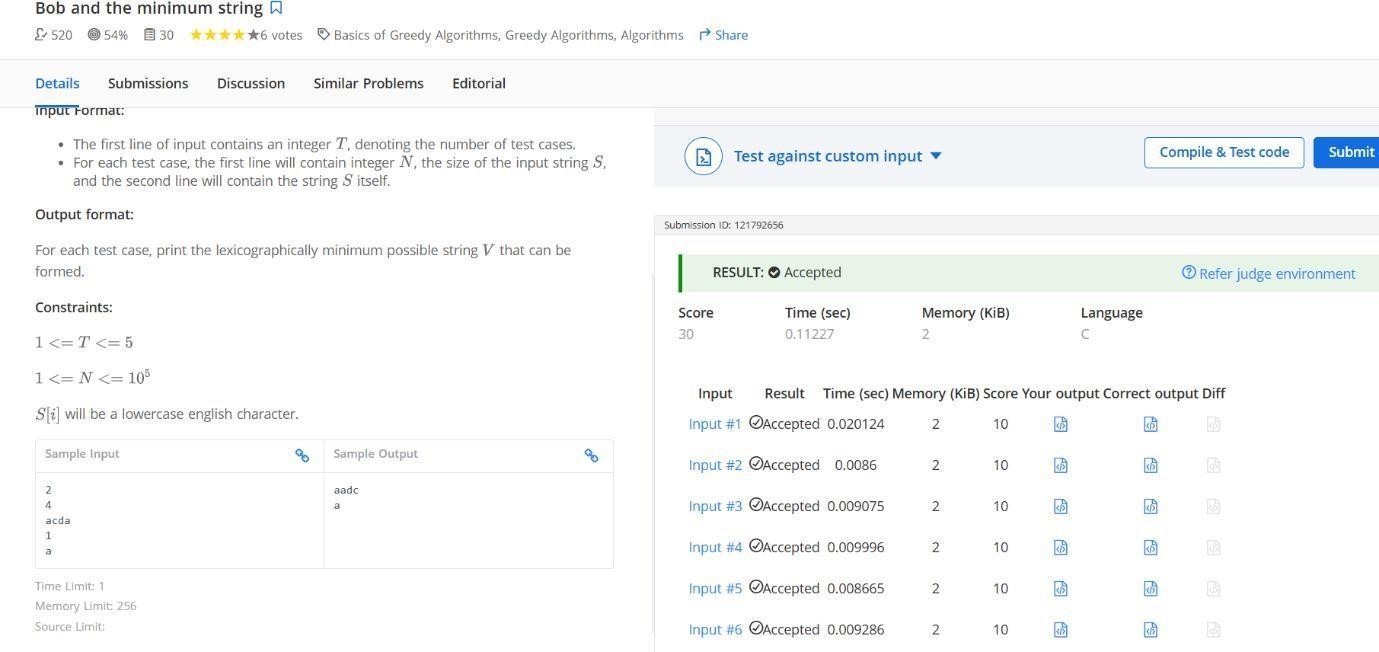
while (top >= 0) result[resIndex++] = stack[top--];

result[resIndex] = '\0'; printf("%s\n", result);

}

return 0;

}



Q.4.

# Code:

import java.util.Scanner;

public class DecreasingPaths {

static int N; static int[][] mat; static long[][] dp; static final int MOD = 1000000007;

static int[] dx = {1, -1, 0, 0};

static int[] dy = {0, 0, 1, -1};

static long dfs(int x, int y) {

if (dp[x][y] != -1) return dp[x][y]; long count = 1; for (int dir = 0; dir < 4; dir++) { int nx = x + dx[dir]; int ny = y + dy[dir];

if (nx >= 0 && nx < N && ny >= 0 && ny < N && mat[nx][ny] < mat[x][y]) { count = (count + dfs(nx, ny)) % MOD;

}

}

dp[x][y] = count; return count;

}

public static void main(String[] args) { Scanner sc = new

Scanner(System.in); N =

sc.nextInt(); mat = new int[N][N]; dp = new long[N][N];

for (int i = 0; i < N; i++) { for (int j = 0; j < N; j++) { mat[i][j] = sc.nextInt();

dp[i][j] = -1;

}

}

long total = 0; for (int i = 0; i < N; i++) { for (int j = 0; j

< N; j++) { total = (total + dfs(i, j)) % MOD;

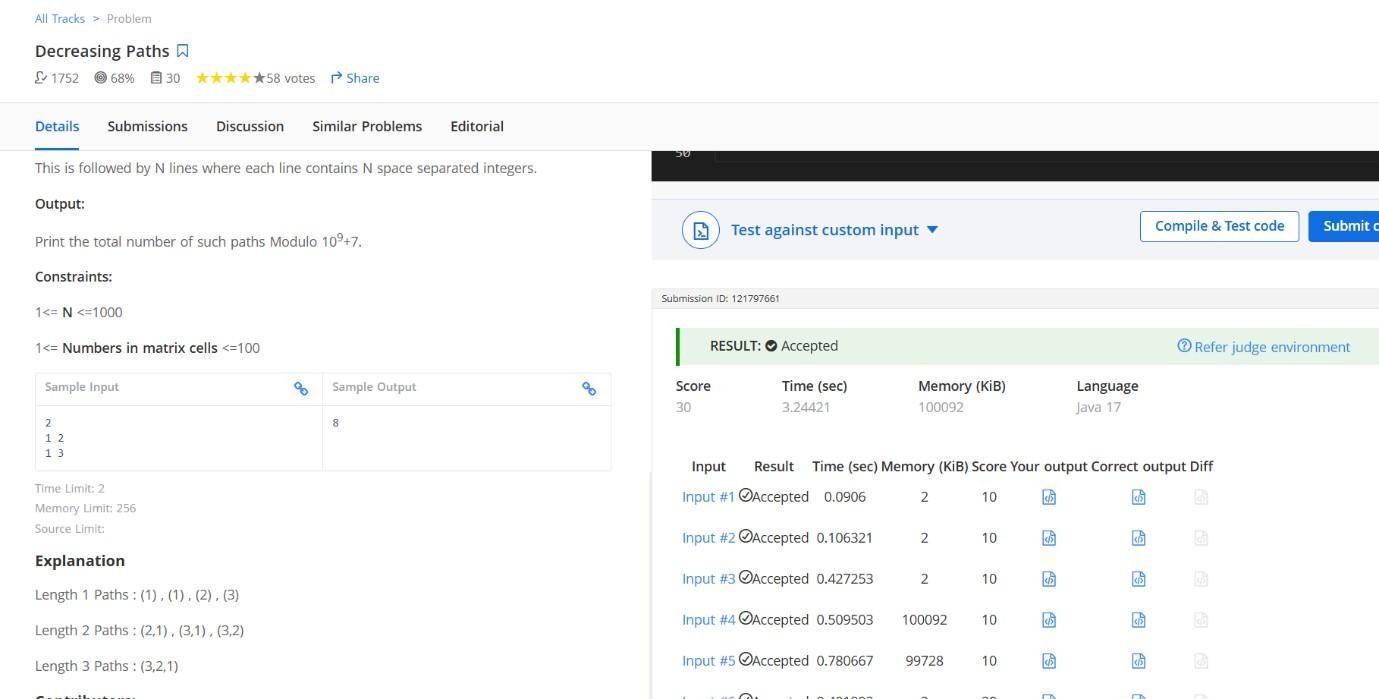
}

}

System.out.println(total); sc.close();

}

}



Q.5

# Code:

#include <stdio.h> #include <stdlib.h> #include <math.h>

#define MOD 1000000007

int used[20]; int n; long long ans = 0;

void dfs(int depth, int prev, int count)

{ if (count > 0) ans = (ans + 1) % MOD;

for (int i = 1; i <= n; i++)

{ if (!used[i]) {

if (prev != -1 && abs(prev - i) == 1) continue;

used[i] = 1; dfs(depth + 1, i, count + 1); used[i] = 0;

}

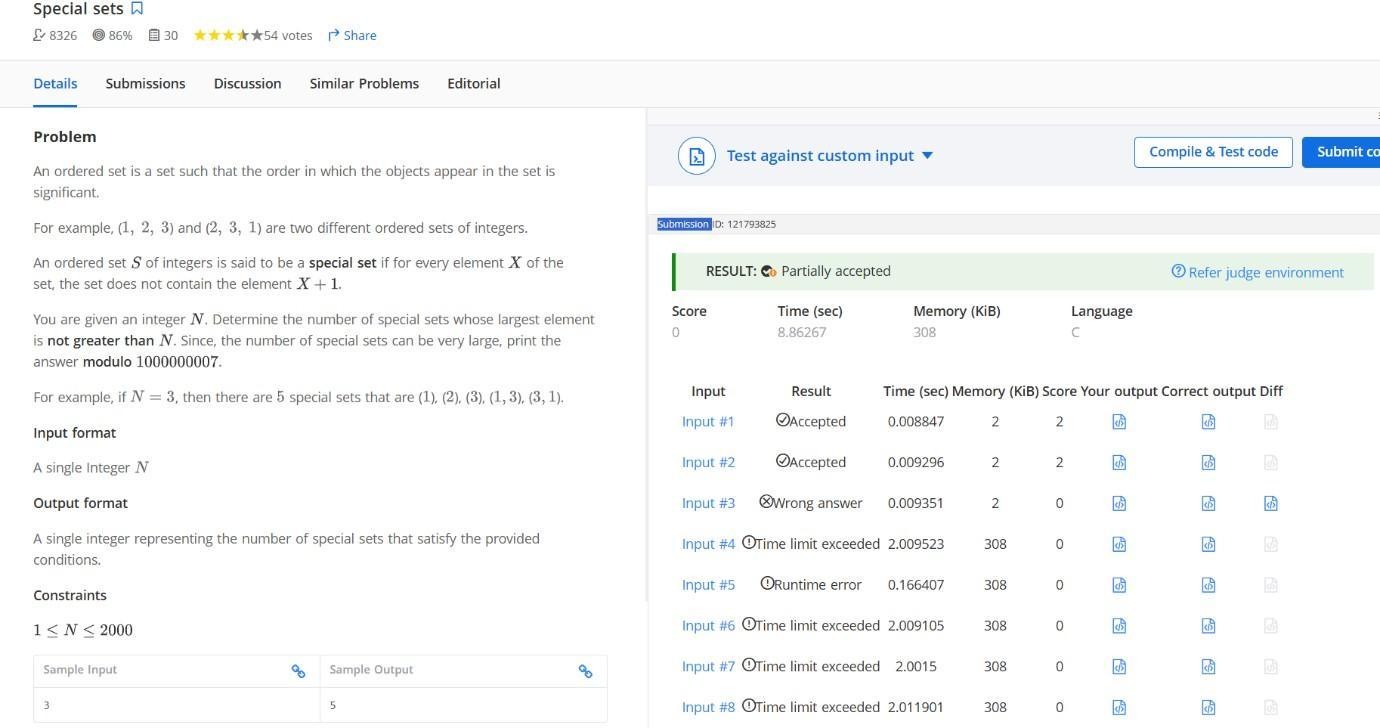
}

}

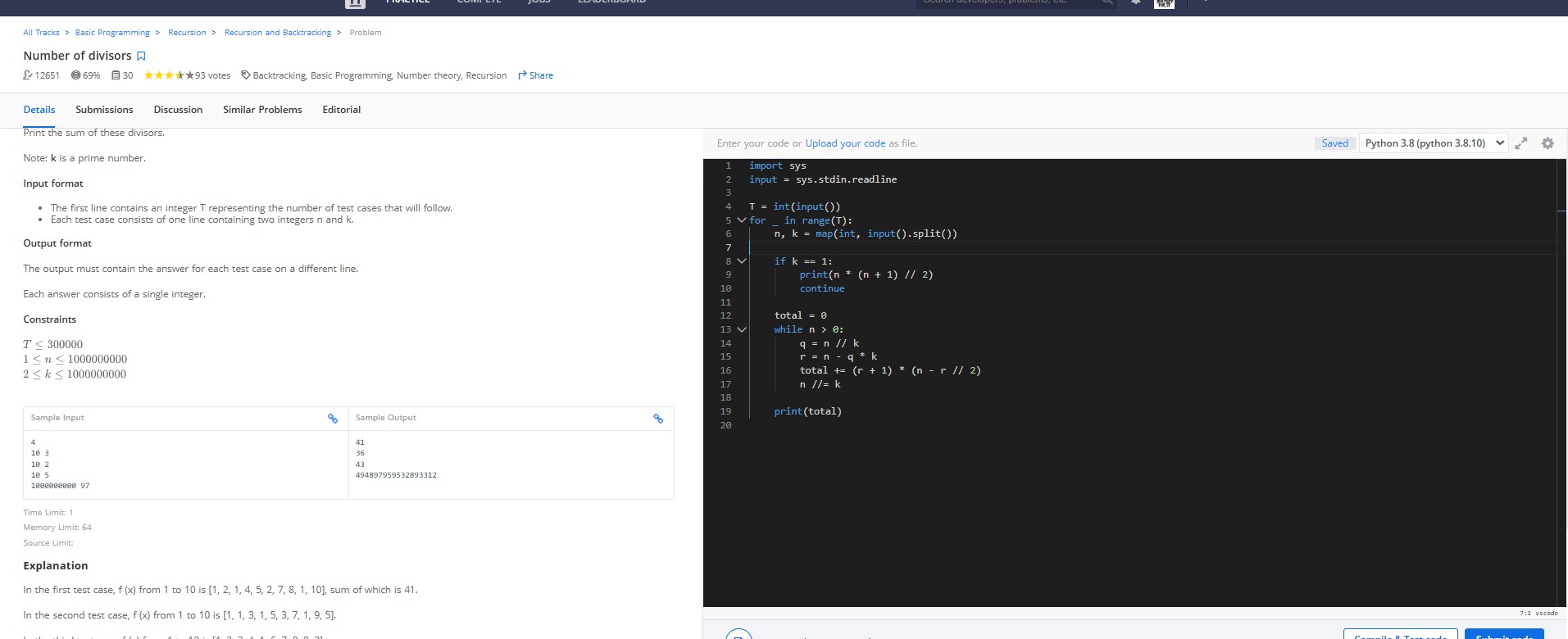
int main() { scanf("%d", &n); ans = 0; for (int i = 0; i <= n; i++) used[i] = 0; dfs(0, -1, 0); printf("%lld\n", ans % MOD);

return 0;

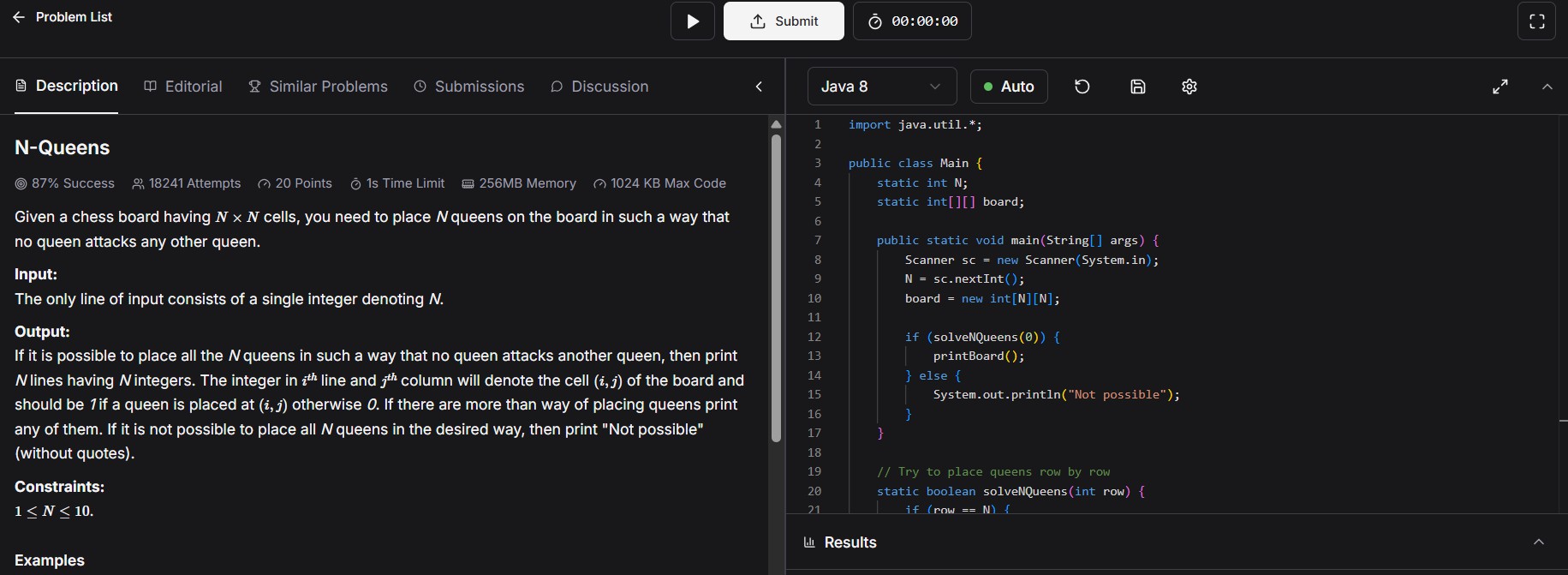
}

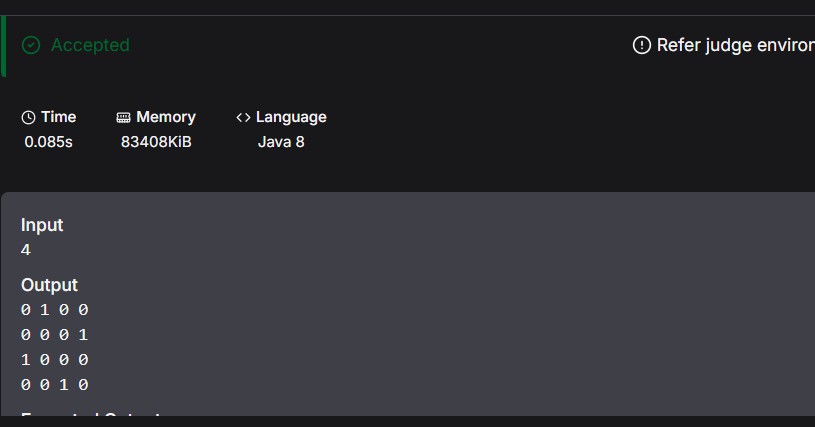


6.

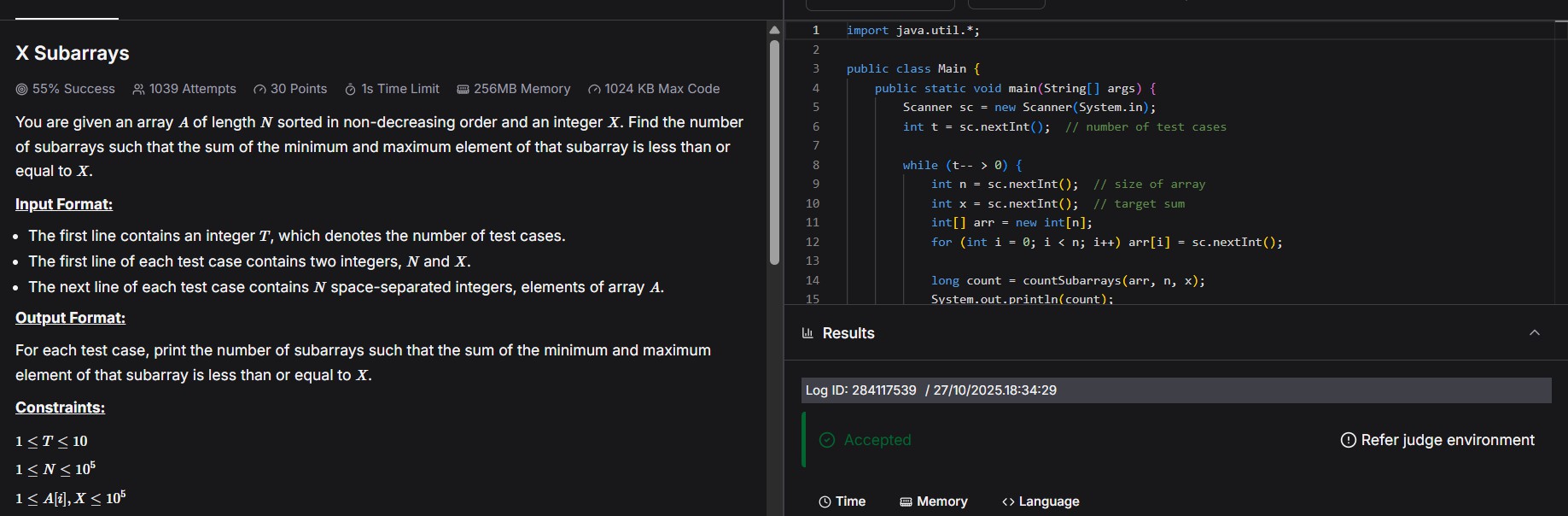


7.

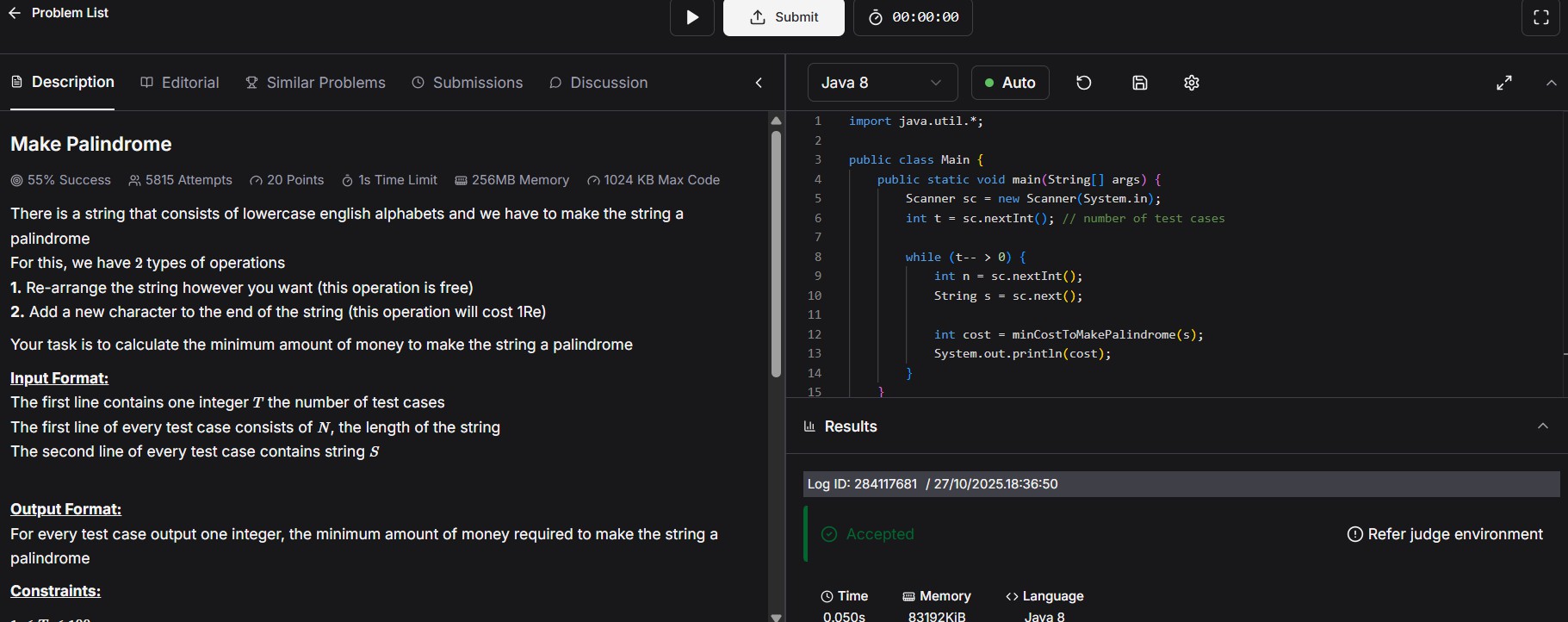




8.



9.



10.

