main.c

Run

Output

Clear

```c
#include <stdio.h>
#include <stdlib.h>

// Calculate modular inverse of a modulo m using brute-force
int mod_inverse(int a, int m) {
    a = a % m;
    for (int x = 1; x < m; x++) {
        if ((a * x) % m == 1)
            return x;
    }
    return -1;
}

// Calculate determinant of 2x2 matrix
int determinant(int m[2][2]) {
    return m[0][0]*m[1][1] - m[0][1]*m[1][0];
}
```

```
Known Plaintext Matrix (P):
[[7, 11],
 [4, 15]]
Known Ciphertext Matrix (C):
[[17, 14],
 [5, 21]]
Inverse of Plaintext Matrix (P⁻¹ mod 26):
[[19, -7],
 [14, 21]]
Recovered Key Matrix (K):
[[25, 19],
 [25, 16]]


=== Code Execution Successful ===
```

main.c

Run

```
            invertible modulo 26.\n");
67          return 1;
68      }
69
70      printf("Inverse of Plaintext Matrix (P⁻¹ mod
            26):\n");
71      print_matrix(P_inv);
72
73      // Compute K = C * P⁻¹ mod 26
74      int K[2][2];
75      multiply_matrices_mod26(C, P_inv, K);
76
77      printf("Recovered Key Matrix (K):\n");
78      print_matrix(K);
79
80      return 0;
81  }
82
```

Output

Clear

```
Known Plaintext Matrix (P):
[[7, 11],
 [4, 15]]
Known Ciphertext Matrix (C):
[[17, 14],
 [5, 21]]
Inverse of Plaintext Matrix (P⁻¹ mod 26):
[[19, -7],
 [14, 21]]
Recovered Key Matrix (K):
[[25, 19],
 [25, 16]]


=== Code Execution Successful ===
```