main.c

Run

Output

Clear

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

// Modular exponentiation: (base^exp) % mod
unsigned long long modexp(unsigned long long base, unsigned long long exp, unsigned long long mod) {
    unsigned long long result = 1;
    base = base % mod;

    while (exp > 0) {
        if (exp % 2 == 1)
            result = (result * base) % mod;

        exp = exp >> 1;
        base = (base * base) % mod;
    }
    return result;
```

```
Ciphertext for 'H': 2369
Attacker recovered plaintext: 7 ('H')


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

main.c

Run

Output

Clear

```
    unsigned long long ciphertext = encrypt
        (plain_char, e, n);
    printf("Ciphertext for 'H': %llu\n",
        ciphertext);

    // Attacker tries to recover it
    int recovered = brute_force_decrypt
        (ciphertext, e, n);
    if (recovered != -1)
        printf("Attacker recovered plaintext: %d
            ('%c')\n", recovered, recovered +
            'A');
    else
        printf("Attack failed.\n");

    return 0;
}
```

```
Ciphertext for 'H': 2369
Attacker recovered plaintext: 7 ('H')


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