main.c

Run

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

#define MAX_LEN 1000
#define ALPHABET_SIZE 26

// English letter frequency ranking (most to least common)
const char english_freq_order[ALPHABET_SIZE + 1] = "ETAOINSHRDLCUMWFGYPBVKJXQZ";

typedef struct {
    char letter;
    int count;
} Frequency;

```

Output

Clear

```
Enter the ciphertext (uppercase letters only):
ZOL SLHY AV AOL NYA
Enter number of top plaintexts to show: 3

[Guess 1] RAT STIO EH EAT NOE

[Guess 2] RAE SEIO TH TAE NOT

[Guess 3] ROE SEIA TH TOE NAT


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

main.c

Run

Output

Clear

```
16
17  // Count letter frequencies
18  void countFrequencies(const char *text,
        Frequency freq[]) {
19      for (int i = 0; i < ALPHABET_SIZE; i++) {
20          freq[i].letter = 'A' + i;
21          freq[i].count = 0;
22      }
23
24      for (int i = 0; text[i]; i++) {
25          if (isupper(text[i])) {
26              freq[text[i] - 'A'].count++;
27          }
28      }
29  }
30
31  // Sort by frequency (descending)
32  int compareFreq(const void *a, const void *b) {
```

```
Enter the ciphertext (uppercase letters only):
ZOL SLHY AV AOL NYA
Enter number of top plaintexts to show: 3

[Guess 1] RAT STIO EH EAT NOE

[Guess 2] RAE SEIO TH TAE NOT

[Guess 3] ROE SEIA TH TOE NAT


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

main.c

Run

```
        (basic approach)
    if (i > 0) {
        swapLetters(map, 'E', 'T');
        if (i > 1) swapLetters(map, 'A',
            'O');
        if (i > 2) swapLetters(map, 'H',
            'N');
    }

    decryptWithMap(ciphertext, map,
        plaintext);
    printf("\n[Guess %d] %s\n", i + 1,
        plaintext);
}

return 0;
}
```

Output

Clear

```
Enter the ciphertext (uppercase letters only):
ZOL SLHY AV AOL NYA
Enter number of top plaintexts to show: 3

[Guess 1] RAT STIO EH EAT NOE

[Guess 2] RAE SEIO TH TAE NOT

[Guess 3] ROE SEIA TH TOE NAT


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