Q2:

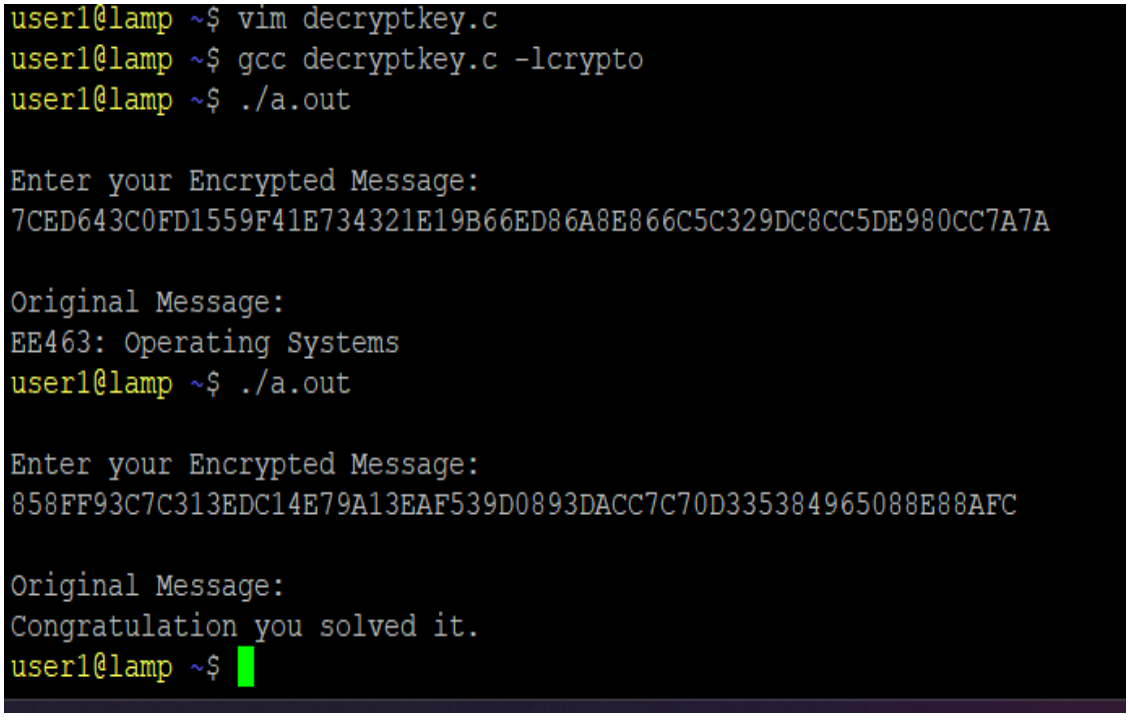
code:

```
#include <stdio.h>
#include <string.h>
#include <openssl/bn.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

void printBN(char *msg, BIGNUM *tmp) {
   char *number_str = BN_bn2hex(tmp); // Convert BIGNUM to hex
   printf("%s%s\n", msg, number_str); // Print hex
   OPENSSL_free(number_str); // Free memory
}

int main(int argc, char *argv[]) {
   BN_CTX *ctx = BN_CTX_new();
   BIGNUM *e = BN_new(); // Encryption Key variable
   BIGNUM *d = BN_new(); // Decryption Key variable
   BIGNUM *n = BN_new(); // Product of large prime numbers p and q
   BIGNUM *phin = BN_new(); // Totient of (n) Euler's totient function
   BIGNUM *C = BN_new(); // Encrypted Message variable
   BIGNUM *P = BN_new(); // Decrypted Ciphertext variable

   // Find Decryption Key (d) using (e) and (Phin):
   // 1- Assign value to (e) Encryption Key from hex
   BN_hex2bn(&e, "010001");
```

```
    // 2- Assign value to (Phin) Encryption Key from hex
    BN_hex2bn(&phin,
"E103ABD94892E3E74AFD724BF28E78348D52298BD687C44DEB3A81065A7981A4");

    // 3- Calculate the Decryption Key (Private Key) d = e mod(Phi(n))
    BN_mod_inverse(d, e, phin, ctx);

    // Read the Encrypted Message from the user to variable CC
    char CC[100];
    printf("\nEnter your Encrypted Message:\n");
    scanf("%s", CC);

    // Assign the input value in variable (CC) to Encrypted Message variable
    BN_hex2bn(&C, CC);

    // Assign value to (n) product of two large prime numbers from hex
    BN_hex2bn(&n,
"E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1");

    // Decrypt Ciphertext using the Private Key
    BN_mod_exp(P, C, d, n, ctx);

    // Convert Hex string to ASCII letters
    printf("\nOriginal Message:\n");
    char str1[500] = "print(\"";
    char *str2 = BN_bn2hex(P);
    char str3[] = "\".decode(\"hex\"))";
    strcat(str1, str2);
    strcat(str1, str3);

    char* args[] = {"python2", "-c", str1, NULL};
    execvp("python2", args);
```

```
    // Free allocated memory
    BN_free(e);
    BN_free(d);
    BN_free(n);
    BN_free(phin);
    BN_free(C);
    BN_free(P);
    BN_CTX_free(ctx);

    return EXIT_SUCCESS;
}
```

Screenshot of output:



Dissection:

This C program aims to find the private key and decrypt a given encrypted message. It utilizes the OpenSSL library to perform the necessary cryptographic operations. The program begins by initializing the required variables, including the encryption key, decryption key, product of prime numbers, totient of 'n', encrypted message, and decrypted ciphertext. The decryption key is calculated using the encryption key and the totient value. Then, the user is prompted to enter the encrypted message, which is stored in the 'CC' variable. The program proceeds to decrypt the ciphertext using the private key and the product of prime numbers. Finally, the decrypted message is converted from a hexadecimal string to ASCII characters and printed as the original message using a Python command executed through the execvp function.