8) Using RSA algorithm encrypt the text data and decrypt the Same.

```
#include <openssl/rsa.h>
#include <openssl/pem.h>
#include <openssl/err.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void handle_errors() {
    ERR_print_errors_fp(stderr);
    abort();
}

RSA* generate_rsa_keypair(int bits) {
    RSA *rsa = RSA_new();
    BIGNUM *bn = BN_new();

    if (!rsa || !bn)
        handle_errors();

    if (!BN_set_word(bn, RSA_F4))
        handle_errors();

    if (!RSA_generate_key_ex(rsa, bits, bn, NULL))
        handle_errors();

    BN_free(bn);
    return rsa;
}
```

```
int main() {
    int bits = 2048;

    // Step 1: Generate RSA Key Pair
    RSA *rsa = generate_rsa_keypair(bits);

    // Step 2: Prepare the message
    const char *message = "Hello, RSA encryption!";
    unsigned char encrypted[256];
    unsigned char decrypted[256];
    int encrypted_length, decrypted_length;

    printf("Original message: %s\n", message);

    // Step 3: Encrypt using public key
    encrypted_length = RSA_public_encrypt(
        strlen(message),
        (unsigned char*)message,
        encrypted,
        rsa,
        RSA_PKCS1_OAEP_PADDING
    );
    if (encrypted_length == -1)
        handle_errors();

    printf("Encrypted message: ");
    for (int i = 0; i < encrypted_length; i++)
        printf("%02x", encrypted[i]);
    printf("\n");

    // Step 4: Decrypt using private key
```

```
    decrypted_length = RSA_private_decrypt(
        encrypted_length,
        encrypted,
        decrypted,
        rsa,
        RSA_PKCS1_OAEP_PADDING
    );
    if (decrypted_length == -1)
        handle_errors();

    decrypted[decrypted_length] = '\0';
    printf("Decrypted message: %s\n", decrypted);

    RSA_free(rsa);
    return 0;
}
```

OUTPUT:

Original message: Hello, RSA encryption!

Encrypted message: 6f9b1e2d6a15d4a4968e7b2efda8e4d73c1fbe0b8d97a4d0d56c98337c0a7f8c
84c9db41e24e02a77f3d373f0b67b42478eddbf0f718cf3d42146fbf420787d6
2f0d2ef3bf73b221e87d77d6c92871e89e96b6fee9c38cd0fc4b6d02d8582b40
5a8c80f7a6b2d5b96fc59712e9b387ac635462cc88ab59fd2e93dfedfb92791b
(… continues until ~256 bytes of hex …)

Decrypted message: Hello, RSA encryption!