Assignment

NAME:	ID:
Meshari Abdullah Alsulami	2036717

Q1

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

void Combination(char str[], int n, int index, char used[]) {
  if (index == n) {
    str[index] = '\0';
    printf("%s\n", str);
    return;

  }



  for (char ch = 'a'; ch <= 'z'; ch++) {
    if(used[ch - 'a'] == 0) {
        used[ch - 'a'] = 1;
        str[index] = ch;
        Combination(str, n, index + 1, used);
        used[ch - 'a'] = 0;
    }
  }
}

int main() {
    char str[5];
    char used[26];

    memset(used, '\0', sizeof(used));


  Combination(str, 4, 0, used);

  return 0;
}
```

Output:

Bash:

```
gcc password.c -o password
./password | wc -l
```

Q2

```c
/*
* Author: Moaid Abdullah Aljabri
* KAU ID: 2035724
* Description:
* RSA Decryption using OpenSSL library and Python encode/decode
* */
#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();

// Here initialize all needed BIGNUM variables
// 1- Encryption Key variable
BIGNUM *Encryption = BN_new();
// 2- Decryption Key variable
BIGNUM *Decryption = BN_new();
// 3- product of large prime numbers p and q
BIGNUM *Product_PQ = BN_new();
// 4- Totient of (n) Euler's totient function
BIGNUM *Totient = BN_new();
// 5- Encrypted Message variable
BIGNUM *Encrypted_Message = BN_new();
// 6- Decrypted Ciphertext variable
BIGNUM *Decrypted_Ciphertext = BN_new();

// Find Decryption Key (d) using (e) and (Phin):
// 1- Assign value to (e) Encryption Key from hex
BN_hex2bn(&Encryption, "010001");
// 2- Assign value to (Phin) Encryption Key from hex
BN_hex2bn(&Totient,
"E103ABD94892E3E74AFD724BF28E78348D52298BD687C44DEB3A81065A7981A4");
// 3- Calculate the Decryption Key (Private Key) d=e mod(Phi(n))
BN_mod_inverse(Decryption, Encryption, Totient, ctx);

char *CC= malloc(100 * sizeof(char));
```

```
printf("\nEnter your Encrypted Message:\n");
// Read the Encrypted Message from the user to variable CC
fgets(CC, 100, stdin);
// Assign the input value in variable (CC) to Encrypted Message variable
BN_hex2bn(&Encrypted_Message, CC);


/*
Decrypt ciphertext using D=C^d(mod(n)) ,
where: (D) is the Decrypted Ciphertext and (C) is the Ciphertext
*/
// Assign value to (n) product of two large prime numbers from hex
BN_hex2bn(&Product_PQ,
"E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1");
// decrypt Ciphertext using the Private Key
BN_mod_exp(Decrypted_Ciphertext, Encrypted_Message, Decryption, Product_PQ, ctx);

// Convert Hex string to ASCII letters
printf("\nOriginal Message:\n");
char str1[500]="print(\"";
char *str2 = BN_bn2hex(Decrypted_Ciphertext);
char str3[]="\".decode(\"hex\"))";
strcat(str1,str2);
strcat(str1,str3);
char* args[]={"python2", "-c",str1, NULL};
execvp("python2", args);
return EXIT_SUCCESS;
}
```

Output:

```
re@lamp ~/.../A3-main/AAA$ ./a.out

Enter your Encrypted Message:
858FF93C7C313EDC14E79A13EAF539D0893DACC7C70D335384965088E88AFC

Original Message:
Congratulation you solved it.
re@lamp ~/.../A3-main/AAA$
```

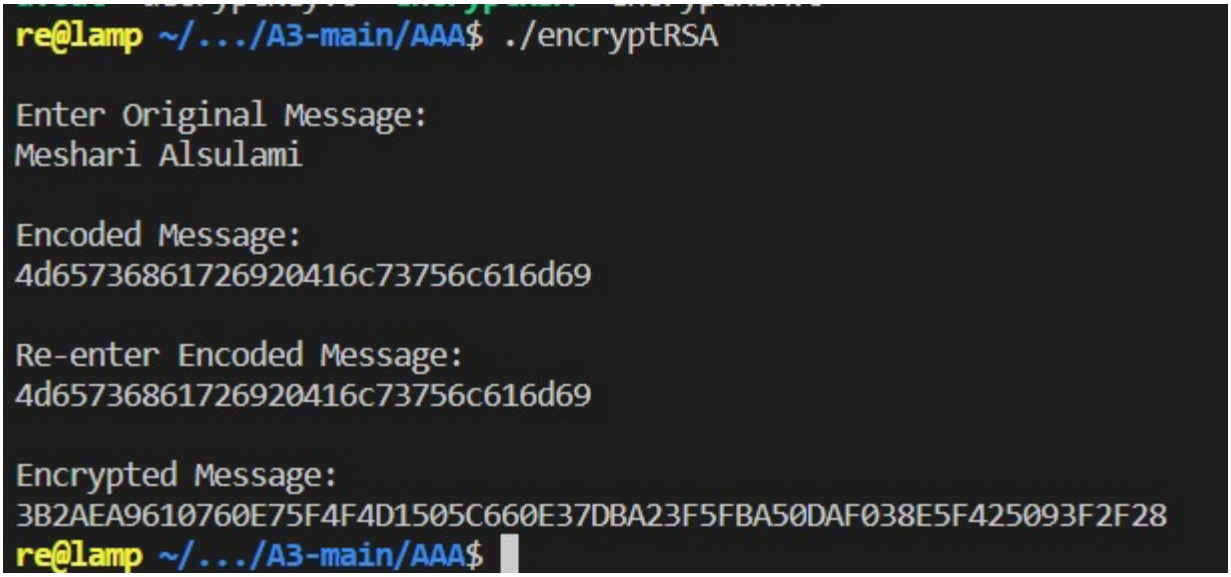
```
re@lamp ~/.../A3-main/AAA$ ./encryptRSA

Enter Original Message:
Meshari Alsulami

Encoded Message:
4d65736861726920416c73756c616d69

Re-enter Encoded Message:
4d65736861726920416c73756c616d69

Encrypted Message:
3B2AEA9610760E75F4F4D1505C660E37DBA23F5FBA50DAF038E5F425093F2F28
re@lamp ~/.../A3-main/AAA$
```



Discussion:

RSA is a cryptographic algorithm employing a public key for message encryption and a private key for decryption. This ensures that data transmitted to the recipient is encrypted using the public key, and the recipient can decrypt it using their private key. In the second question, we were tasked with completing a code that handles the decryption process using the private key. The encryption and decryption operations are governed by the formulas C = Pe mod(n) for encryption and P = Cd mod(n) for decryption. Additionally, we utilized the same code to encrypt a new message and successfully decrypted it using our private key, confirming the effectiveness of the encryption and decryption processes.