



# EE 463 – Operating system

## Semester 1 2023/2024

**Title:** Lap exam

**Student name:** Moaid Abdullah Aljabri

**ID:** 2035724

# Q1)

## Source code:

```
// Moaid Abdullah Aljabri
// 2035724

#include <stdio.h>

int main (int argc, char* argv[]){

    char letter[] = "abcdefghijklmnopqrstuvwxyz"; // array of all letters in
english
    int l1; // letter 1
    int l2; // letter 2
    int l3; // letter 3
    int l4; // letter 4

    for(l1 = 0; l1 < 26; l1++){ // for loop for the first letter to go from a to
z
    for(l2 = 0; l2 < 26; l2++){ // second for loop for the second letter to go
from a to z
        if(l2 == l1 || l2 == l3 || l2 == l4 ){
            // if letter 2 is the same as any letter it will skip the next
iteration
            continue;
        }
    for(l3 = 0; l3 < 26; l3++){ // third for loop for the third letter to go from
a to z
        if(l3 == l1 || l3 == l2 || l3 == l4 ){
            // if letter 3 is the same as any letter it will skip the next
iteration
            continue;
        }
    for(l4 = 0; l4 < 26; l4++){ // forth for loop for the forth letter to go from
a to z
        if(l4 == l1 || l4 == l2 || l4 == l3 ){
            // if letter 4 is the same as any letter it will skip the next
iteration
            continue;
        }
        // print the result inside loop 4
        printf("%c%c%c%c\n",letter[l1],letter[l2],letter[l3],letter[l4]);
    }
    }
```

```
    }
    }



    return 0;
}
```

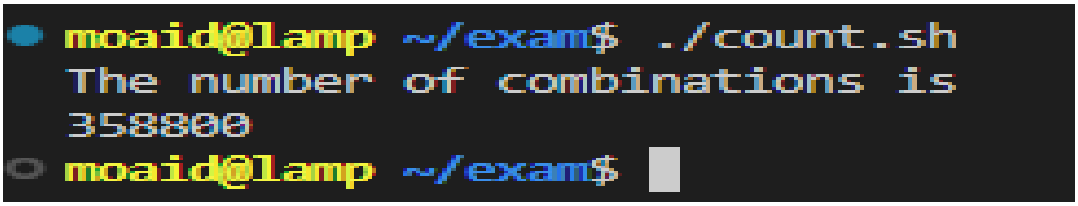
## Output:

```
zyvp
zyvq
zyvr
zyvs
zyvt
zyvu
zyvw
zyvx
zywa
zywb
zywc
zywd
zywe
zywf
zywg
zywh
zywi
zywj
zywk
zywl
zywm
zywn
zywo
zywp
zywq
zywr
zyws
zywt
zywu
zywv
zywx
zyxa
zyxb
zyxc
zyxd
zyxe
zyxf
zyxg
zyxh
zyxi
zyxj
zyxk
zyxl
zyxm
zyxn
zyxo
zyxp
zyxq
zyxr
zyxs
zyxt
zyxu
zyxv
zyxw
moaid@lamp ~/exam$
```

**Bash code:**

```
echo "The number of combinations is "
./pass | wc -l
```

**Output:**



# Q2)

## Source code:

```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:**

```
moaid@lamp ~/exam$ ./a.out

Enter your Encrypted Message:
858FF93C7C313EDC14E79A13EAF539D0893DACC7C70D335384965088E88AFC

Original Message:
Congratulation you solved it.
moaid@lamp ~/exam$ ./encryptRSA

Enter Original Message:
Moaid Abdullah Aljabri

Encoded Message:
4d6f616964204162647 56c6c616820416c6a61627269

Re-enter Encoded Message:
4d6f616964204162647 56c6c616820416c6a61627269

Encrypted Message:
D7F80CDE62ADB5B7A4912DE689C8861F9C3E627C61CA9A3FDDB910A455B5A5E4
moaid@lamp ~/exam$ ./a.out

Enter your Encrypted Message:
D7F80CDE62ADB5B7A4912DE689C8861F9C3E627C61CA9A3FDDB910A455B5A5E4

Original Message:
Moaid Abdullah Aljabri
moaid@lamp ~/exam$
```

## Discussion:

RSA is an algorithm that uses a public key to encrypt a message and private key to decrypt the message so that any data sent to the client is encrypted by the public key and client knows the decrypted key. In the second question in the exam, we are requested to complete a code that decrypts the message (private key). We use these formulas $C = P^e \bmod(n)$ to encrypt the data and $P = C^d \bmod(n)$ to decrypt the message. We decrypted the message in task 2 and it says, “Congratulation you solved it”. Also, we used the coder to encrypt a new message and decrypted with our private key as you can see in the output.