**CRYPTOGRAPHY CODING ASSIGNMENT**

**SANSKRATI JAIN**

**CSB20047**

**Assignment 2:**

Implement a PRF and PRP by utilizing the random sequence of bytes collected from  either "/dev/random" , OR "/dev/urandom".

**Solution**

Points to note:

* For generating key “/dev/urandom” is used.
* x is of length 16 bits that is 4 characters and k is of 4 bits i.e k = 0, k = 1, k = 2, and k = 3.
* Code of PRF and PRG are almost similar except for the check\_duplicate function. For PRF it make sure that no duplicate key is stored in a column i.e for a particular k every key is different. But in case of PRP every key that is stored is different.
* The length of key is 2 ^ 64.

**Code for generating key:**

#include <stdio.h>

#include <unistd.h>

#include <fcntl.h>

#include <math.h>

int main()

{

    int i;

    unsigned long long int key\_size = pow(2, 64); //key size

    unsigned char buffer[8];

    int fd = open("/dev/urandom", O\_RDONLY);

    read(fd, buffer, key\_size);

    // buffer now contains the random data

    close(fd);

    //write file

    FILE \*fptr;

    fptr = fopen("key.txt", "w");

    for (i = 0; i < key\_size; ++i)

    {

        printf("%02X", buffer[i]);

        fprintf(fptr, "%02X", buffer[i]);

    }

    printf("\n");

    fclose(fptr);

    return 0;

}

**Code for PRF:**

# This is the implementation of Pseudo random function

# Submitted by - CSB20047

# Function to convert Hexadecimal to binary

def hex2bin(s):

    mp = {'0': "0000",

          '1': "0001",

          '2': "0010",

          '3': "0011",

          '4': "0100",

          '5': "0101",

          '6': "0110",

          '7': "0111",

          '8': "1000",

          '9': "1001",

          'A': "1010",

          'B': "1011",

          'C': "1100",

          'D': "1101",

          'E': "1110",

          'F': "1111"}

    bin = ""

    for i in range(len(s)):

        bin = bin + mp[s[i]]

    return bin

# function to check if the key is already used for a particular k

def check\_duplicate(key, table):

    status = False

    for i in table:

        #if key is same as the key present in table

        if(key[0:16] == i[1] or key[16:32] == i[2] or key[32:48] == i[3] or key[48:64] == i[4]):

            status = True

            break

    return status

# function to implement prf

def prf(length):

    # table to store mapping

    table = []

    table.append(["x", "k = 0", "k = 1", "k = 2", "k = 3"])

    print(table)

    counter = 0

    #open key and message file

    with open("message.txt") as m, open("key.txt") as k:

        while counter < length:

            # temperory table

            temp\_table = []

            # read 16bits of message

            c = m.read(4)

            if not c:

                print("End of file")

                break

            temp\_table.append(c)

            # read 48 bits of key for k=0, k=1, k=2 and k=3

            key = k.read(16)

            key = hex2bin(str(key))

            #check if the key is duplicate for particular key

            # True if key is duplicate else false

            status = check\_duplicate(key, table)

            # if key is unique

            if(status == False):

                # append key for all k in a row

                temp\_table.append(key[0:16])

                temp\_table.append(key[16:32])

                temp\_table.append(key[32:48])

                temp\_table.append(key[48:64])

                counter += 1

                # print row

                print(temp\_table)

                table.insert(0,temp\_table)

            # key is duplicate

            else:

                continue

# main function

if \_\_name\_\_ == "\_\_main\_\_":

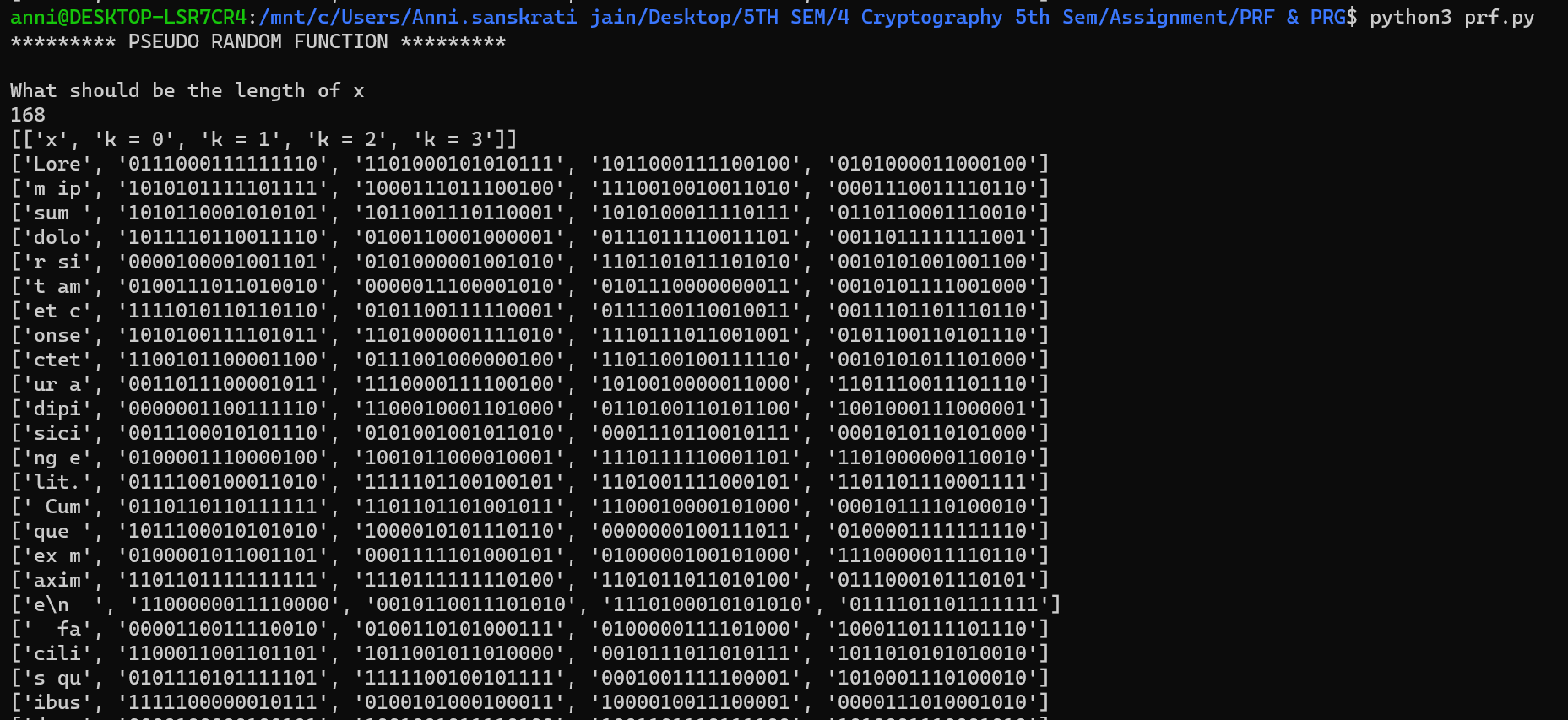
    print("\*\*\*\*\*\*\*\*\* PSEUDO RANDOM FUNCTION \*\*\*\*\*\*\*\*\*")

    print()

    length = int(input("What should be the length of x\n"))

    prf(length)

**Output of PRF:**



**Code for PRP**

# This is the implementation of Pseudo random permutation

# Submitted by - CSB20047

# Function to convert Hexadecimal to binary

def hex2bin(s):

    mp = {'0': "0000",

          '1': "0001",

          '2': "0010",

          '3': "0011",

          '4': "0100",

          '5': "0101",

          '6': "0110",

          '7': "0111",

          '8': "1000",

          '9': "1001",

          'A': "1010",

          'B': "1011",

          'C': "1100",

          'D': "1101",

          'E': "1110",

          'F': "1111"}

    bin = ""

    for i in range(len(s)):

        bin = bin + mp[s[i]]

    return bin

# function to check if the key is duplicate

def check\_duplicate(key, table):

    status = False

    for i in table:

        used\_key = i[1]

        used\_key += str(i[2])

        used\_key += str(i[3])

        used\_key += str(i[4])

        if(key == used\_key):

            status = True

            break

    return status

# function to implement prp

def prp(length):

    # table to store mapping

    table = []

    table.append(["x", "k = 0", "k = 1", "k = 2", "k = 3"])

    print(table)

    counter = 0

    #open key and message file

    with open("message.txt") as m, open("key.txt") as k:

        while counter < length:

            # temperory table

            temp\_table = []

            # read 16bits of message

            c = m.read(4)

            if not c:

                print("End of file")

                break

            temp\_table.append(c)

            # read 48 bits of key for k=0, k=1, k=2 and k=3

            key = k.read(16)

            key = hex2bin(str(key))

            #check if the key is duplicate

            # True if key is duplicate else false

            status = check\_duplicate(key, table)

            # if key is unique

            if(status == False):

                # append key for all k in a row

                temp\_table.append(key[0:16])

                temp\_table.append(key[16:32])

                temp\_table.append(key[32:48])

                temp\_table.append(key[48:64])

                counter += 1

                # print row

                print(temp\_table)

                table.insert(0,temp\_table)

            # key is duplicate

            else:

                continue

# main function

if \_\_name\_\_ == "\_\_main\_\_":

    print("\*\*\*\*\*\*\*\*\* PSEUDO RANDOM PERMUTATION  \*\*\*\*\*\*\*\*\*")

    print()

    length = int(input("What should be the length of x\n"))

    prp(length)

**Output of PRP:**

