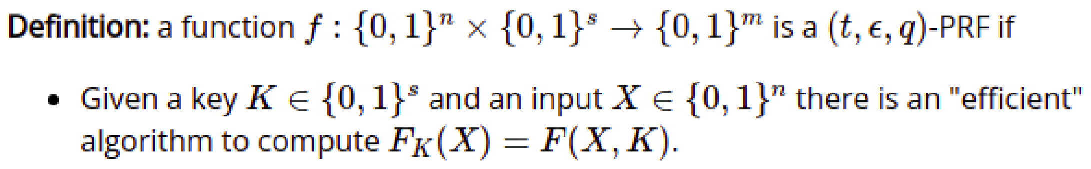
POIS ASSIGNMENT 1

TASK 2

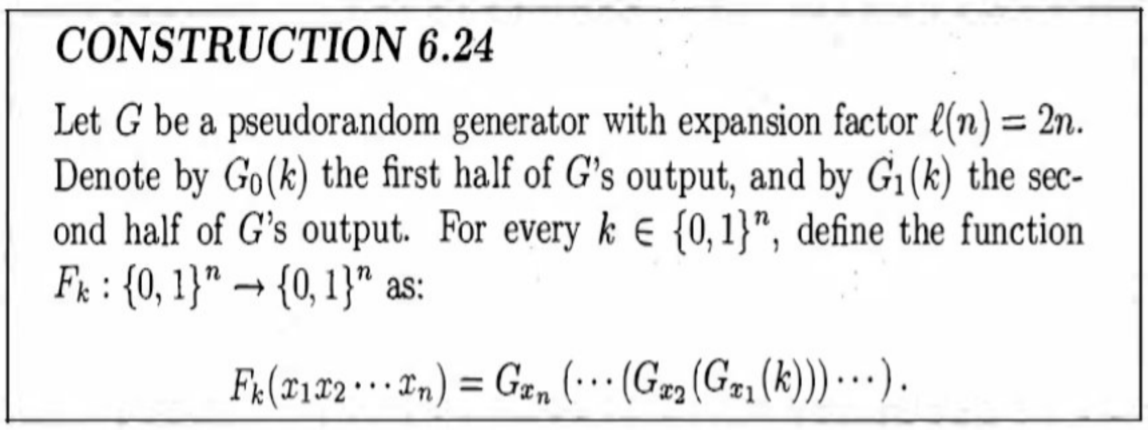
BUILD A PROVABLY SECURE PRG (PSEUDO-RANDOM NUMBER GENERATOR)

CODE



Definition: a function $f : \{0,1\}^n \times \{0,1\}^s \rightarrow \{0,1\}^m$ is a (t, ϵ, q)-PRF if

- Given a key $K \in \{0,1\}^s$ and an input $X \in \{0,1\}^n$ there is an "efficient" algorithm to compute $F_K(X) = F(X, K)$.



CONSTRUCTION 6.24

Let G be a pseudorandom generator with expansion factor $\ell(n) = 2n$. Denote by $G_0(k)$ the first half of G's output, and by $G_1(k)$ the second half of G's output. For every $k \in \{0,1\}^n$, define the function $F_k : \{0,1\}^n \rightarrow \{0,1\}^n$ as:

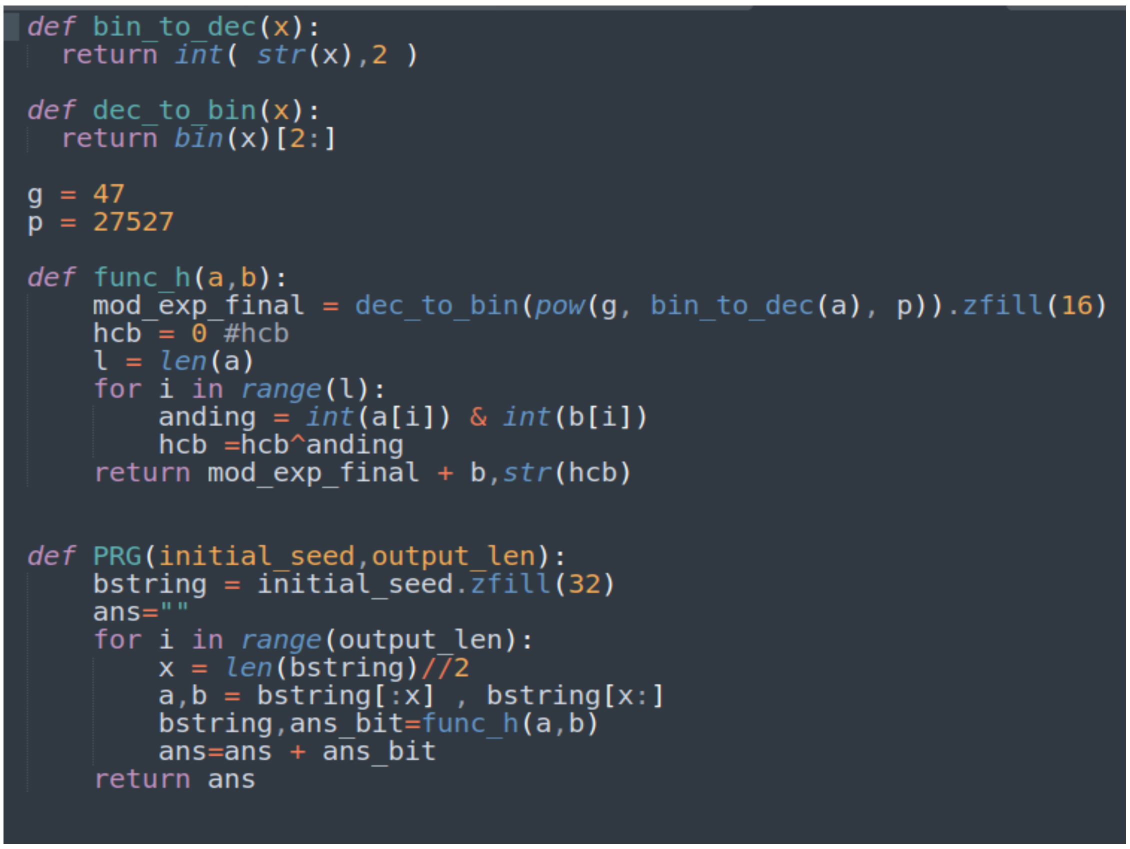
$$F_k(x_1 x_2 \cdots x_n) = G_{x_n}(\cdots(G_{x_2}(G_{x_1}(k)))\cdots).$$

A pseudorandom function from a pseudorandom generator.

The main idea is to imagine a complete binary tree of height in (in will be the input length of the PRF), where each node in the tree represents an application of G.
If a node gets input k, then it sends GL(k) to its left child and sends GR(k) to its right child. There will be 2in leaves, whose values will be the outputs of the PRF.
To access the leaf with index x ∈ {0,1}in. we can traverse the tree from root to leaf, taking left and right turns at each node according to the bits of x.

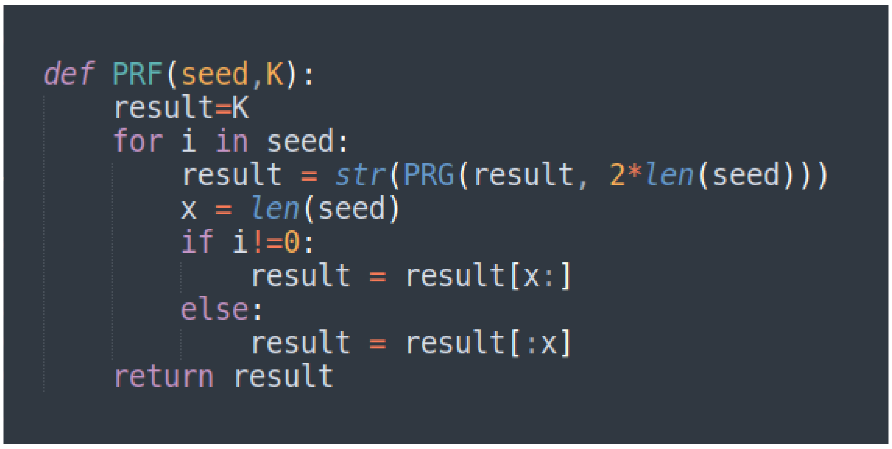
PRG : Function (the PRG itself) that, given a k bit-string in input, outputs a l(k) bit-string and no randomized algorithm can say if the string produced is generated by a real random source or not.

H : Function that helps to find those pseudorandom bit. Since the input of H is a bit string that will be split into two halves, the length of the initial seed must be even.



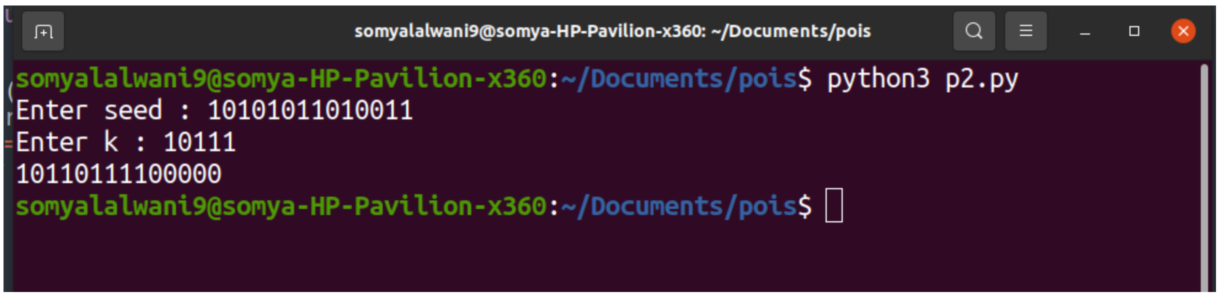
```python
def bin_to_dec(x):
  return int( str(x),2 )

def dec_to_bin(x):
  return bin(x)[2:]

g = 47
p = 27527

def func_h(a,b):
    mod_exp_final = dec_to_bin(pow(g, bin_to_dec(a), p)).zfill(16)
    hcb = 0 #hcb
    l = len(a)
    for i in range(l):
        anding = int(a[i]) & int(b[i])
        hcb =hcb^anding
    return mod_exp_final + b,str(hcb)


def PRG(initial_seed,output_len):
    bstring = initial_seed.zfill(32)
    ans=""
    for i in range(output_len):
        x = len(bstring)//2
        a,b = bstring[:x] , bstring[x:]
        bstring,ans_bit=func_h(a,b)
        ans=ans + ans_bit
    return ans
```

PRF :
The PRG function will return some random output. Based on ith bit of seed, it will be decided, if the left or right half will be selected. If our ith bit of seed is 1, right half; else left half is selected. This becomes the result for the next round.



```python
def PRF(seed,K):
    result=K
    for i in seed:
        result = str(PRG(result, 2*len(seed)))
        x = len(seed)
        if i!=0:
            result = result[x:]
        else:
            result = result[:x]
    return result
```

OUTPUT :



```
somyalalwani9@somya-HP-Pavilion-x360:~/Documents/pois$ python3 p2.py
Enter seed : 10101011010011
Enter k : 10111
10110111100000
somyalalwani9@somya-HP-Pavilion-x360:~/Documents/pois$
```