MAC code explanation:

- This is CBC-MAC construction
- User enters a message and key k is generated randomly
- We will pass this key and message to a function CBC_MAC()
- It will divide message into blocks of length N=16 and padding will be done if it is necessary
- Length of message is encoded into binary and will be prepended to the message
- Now , we will apply block cipher(IV is 0's) on first block(which is length block) and the output will be given to next block cipher
- Like that for every block of message , we apply block cipher and at the end we will return block cipher value of last block.
- Verification of MAC tag will be done by regenerating the MAC tag and will compare with given mac tag.
- verification_CBC_MAC() will take care of above functionality.It will return true/false based on both tags are matching.

```
def CBC_MAC(msg,key,N):
   #N is block length and r is IV(Initialization vector)
   len_msg=len(msg)
   bin_value_len_msg=bin(len_msg).replace('0b','')
   msg_blocks=int(len(msg)/N)
   rem=len(msg)%N

   if rem!=0:
       temp=msg[-1*rem:].zfill(N) # padding last block with 0s and storing
into temp
       msg=msg[:-1*rem] #deleting last block ,which is not having perfect
length 'N'
       msg=msg+temp #adding that last block to msg
   new_msg=bin_value_len_msg+msg #adding msg length to msg

   xor_vec='0'*N  # for xoring with message length block
   #here we need to divide msg into blocks of length 'N'

   for i in range(0,len(new_msg),N):
       msg_bits=new_msg[i:i+N]
       f_ip=bin(int(msg_bits,2)^int(xor_vec,2)).replace('0b','')
       f=PRF(key,f_ip)
       xor_vec=f # this PRF's output is input for next XOR opearation with
next msg_block
```

```
    mac_tag=f #output of PRF's contains final mac tag
    return mac_tag
```

```
def verification_CBC_MAC(msg,key,N,tag):
    re_tag=CBC_MAC(msg,key,N)
    return re_tag==tag
```