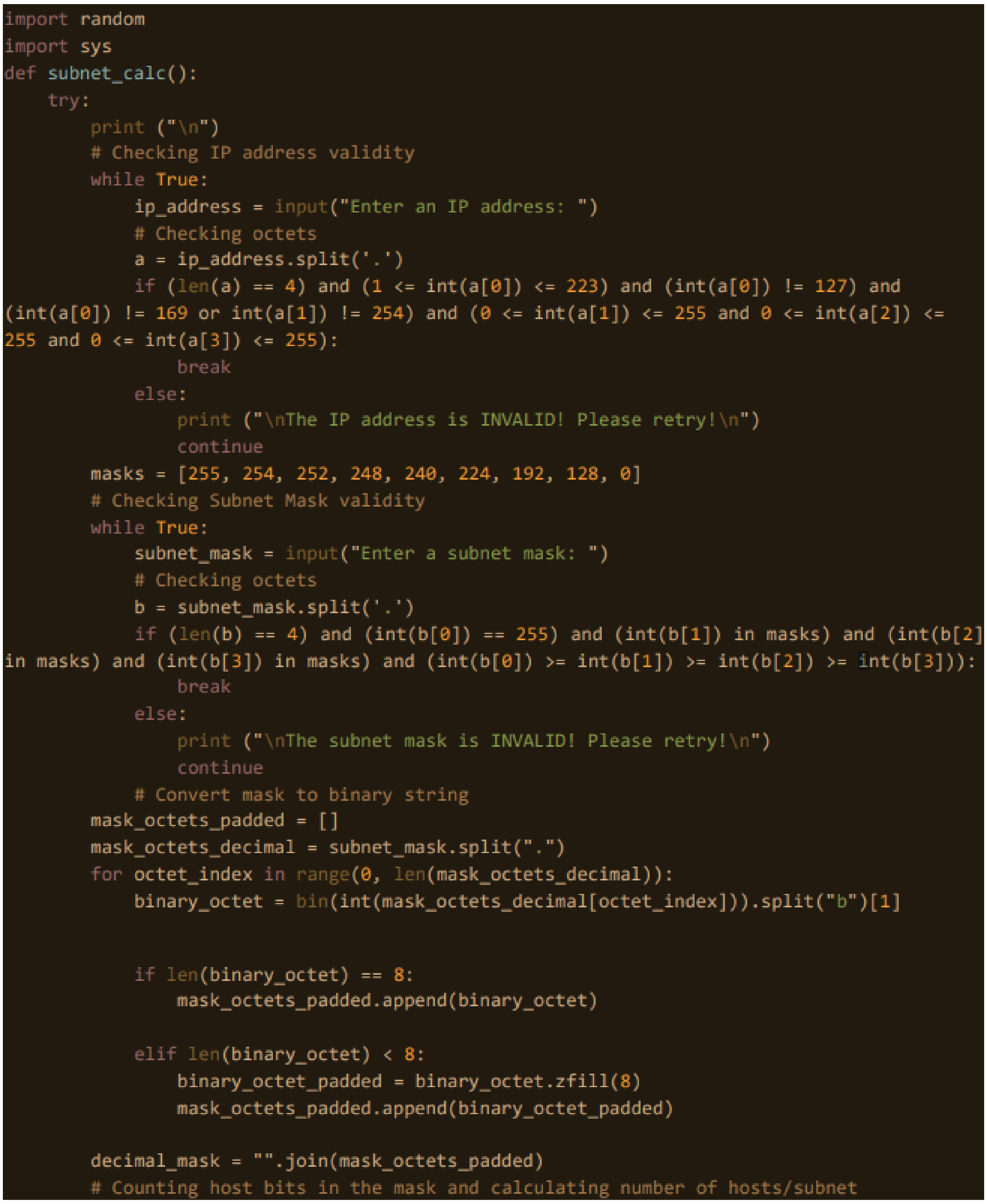
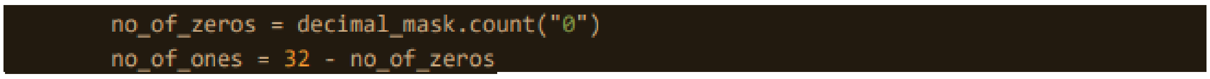
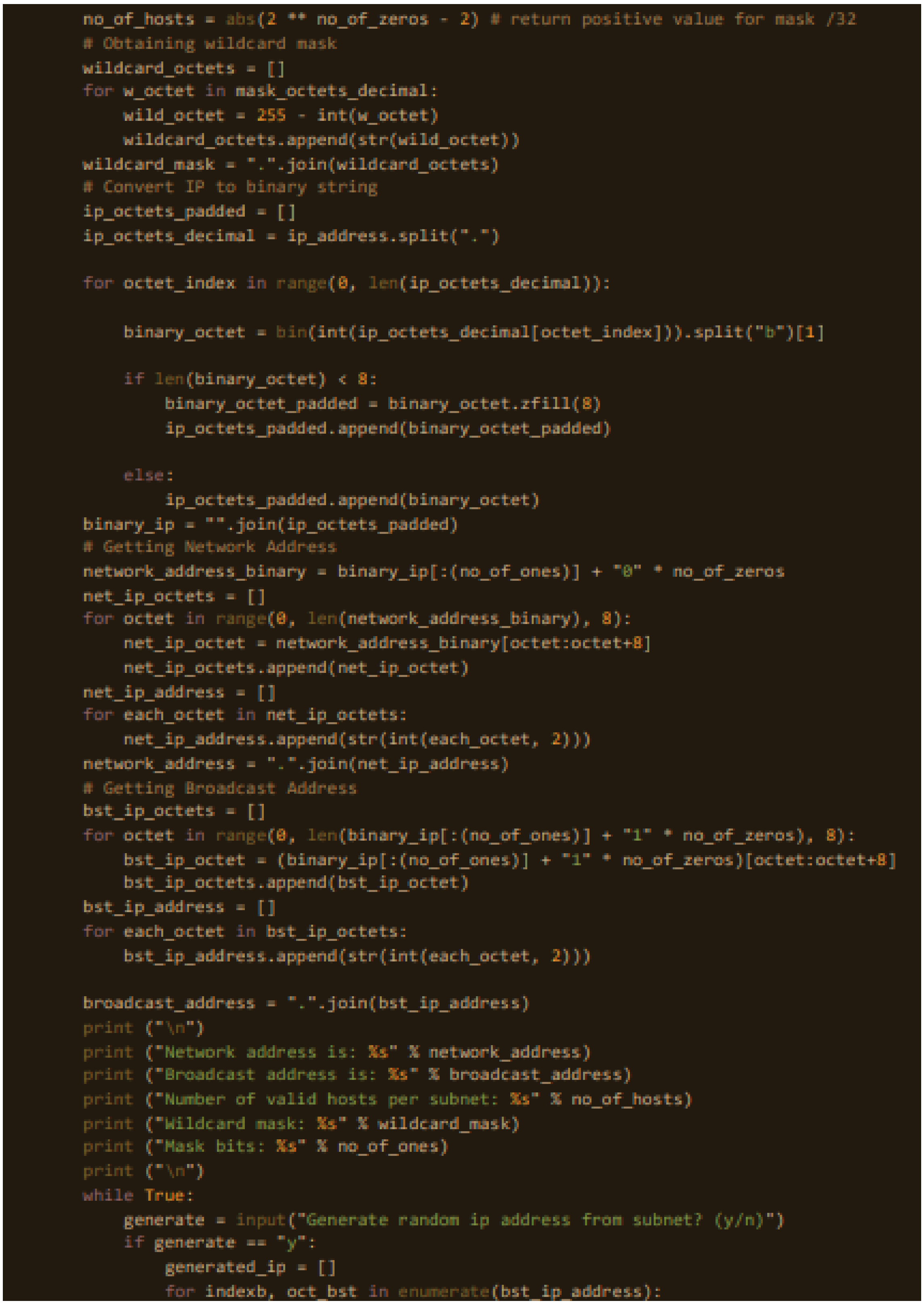
21BCE7371
Radha Krishna Garg



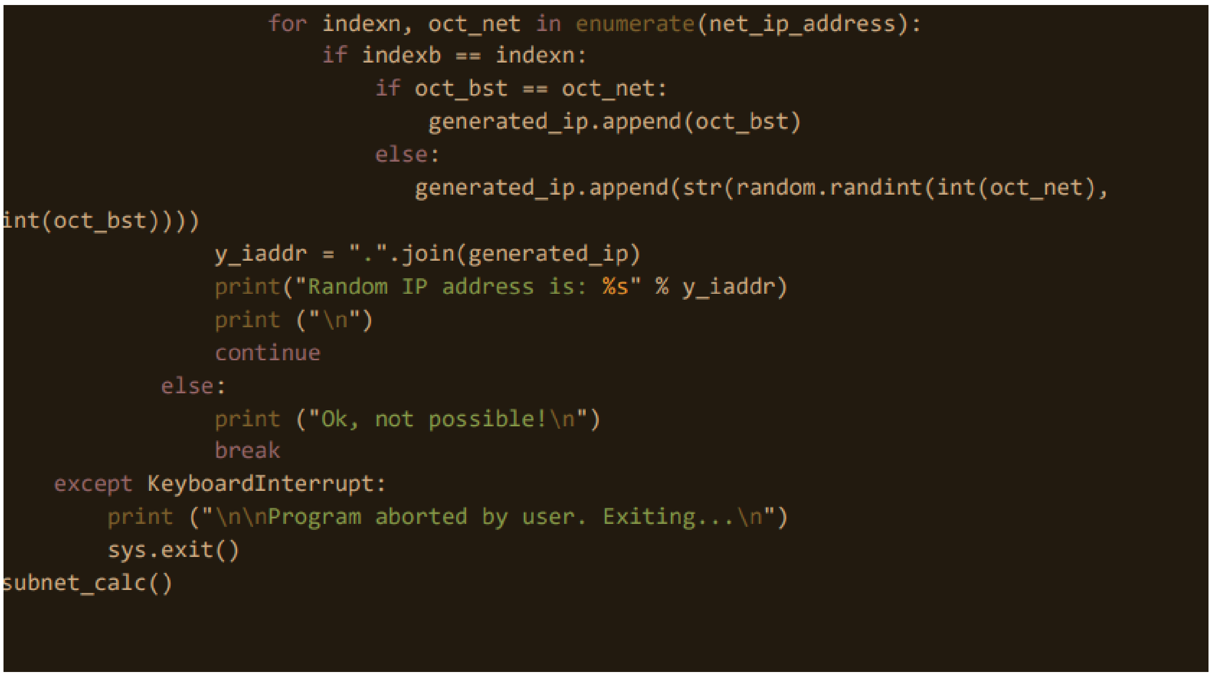
```
import random
import sys
def subnet_calc():
    try:
        print ("\n")
        # Checking IP address validity
        while True:
            ip_address = input("Enter an IP address: ")
            # Checking octets
            a = ip_address.split('.')
            if (len(a) == 4) and (1 <= int(a[0]) <= 223) and (int(a[0]) != 127) and
(int(a[0]) != 169 or int(a[1]) != 254) and (0 <= int(a[1]) <= 255 and 0 <= int(a[2]) <=
255 and 0 <= int(a[3]) <= 255):
                break
            else:
                print ("\nThe IP address is INVALID! Please retry!\n")
                continue
        masks = [255, 254, 252, 248, 240, 224, 192, 128, 0]
        # Checking Subnet Mask validity
        while True:
            subnet_mask = input("Enter a subnet mask: ")
            # Checking octets
            b = subnet_mask.split('.')
            if (len(b) == 4) and (int(b[0]) == 255) and (int(b[1]) in masks) and (int(b[2]
in masks) and (int(b[3]) in masks) and (int(b[0]) >= int(b[1]) >= int(b[2]) >= int(b[3])):
                break
            else:
                print ("\nThe subnet mask is INVALID! Please retry!\n")
                continue
            # Convert mask to binary string
        mask_octets_padded = []
        mask_octets_decimal = subnet_mask.split(".")
        for octet_index in range(0, len(mask_octets_decimal)):
            binary_octet = bin(int(mask_octets_decimal[octet_index])).split("b")[1]


            if len(binary_octet) == 8:
                mask_octets_padded.append(binary_octet)

            elif len(binary_octet) < 8:
                binary_octet_padded = binary_octet.zfill(8)
                mask_octets_padded.append(binary_octet_padded)

        decimal_mask = "".join(mask_octets_padded)
        # Counting host bits in the mask and calculating number of hosts/subnet
```



```
        no_of_zeros = decimal_mask.count("0")
        no_of_ones = 32 - no_of_zeros
```

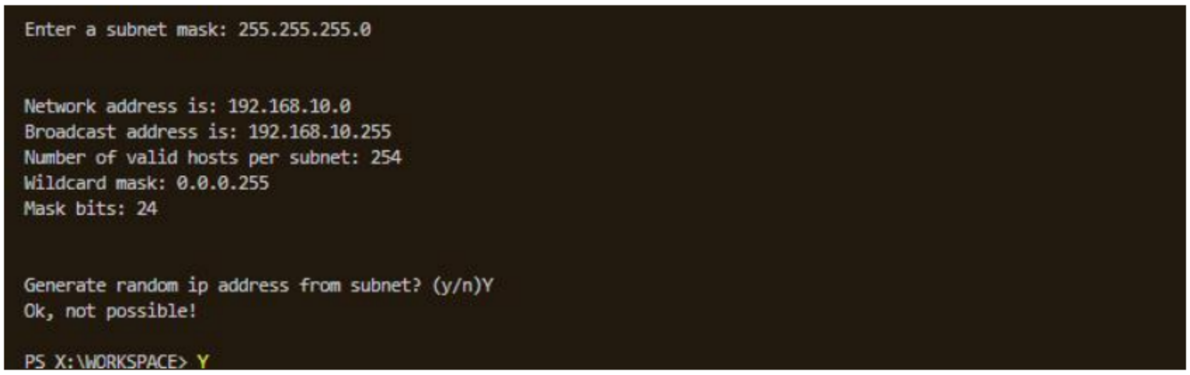


```python
        no_of_hosts = abs(2 ** no_of_zeros - 2) # return positive value for mask /32
        # Obtaining wildcard mask
        wildcard_octets = []
        for w_octet in mask_octets_decimal:
            wild_octet = 255 - int(w_octet)
            wildcard_octets.append(str(wild_octet))
        wildcard_mask = ".".join(wildcard_octets)
        # Convert IP to binary string
        ip_octets_padded = []
        ip_octets_decimal = ip_address.split(".")

        for octet_index in range(0, len(ip_octets_decimal)):

            binary_octet = bin(int(ip_octets_decimal[octet_index])).split("b")[1]

            if len(binary_octet) < 8:
                binary_octet_padded = binary_octet.zfill(8)
                ip_octets_padded.append(binary_octet_padded)

            else:
                ip_octets_padded.append(binary_octet)
        binary_ip = "".join(ip_octets_padded)
        # Getting Network Address
        network_address_binary = binary_ip[:(no_of_ones)] + "0" * no_of_zeros
        net_ip_octets = []
        for octet in range(0, len(network_address_binary), 8):
            net_ip_octet = network_address_binary[octet:octet+8]
            net_ip_octets.append(net_ip_octet)
        net_ip_address = []
        for each_octet in net_ip_octets:
            net_ip_address.append(str(int(each_octet, 2)))
        network_address = ".".join(net_ip_address)
        # Getting Broadcast Address
        bst_ip_octets = []
        for octet in range(0, len(binary_ip[:(no_of_ones)] + "1" * no_of_zeros), 8):
            bst_ip_octet = (binary_ip[:(no_of_ones)] + "1" * no_of_zeros)[octet:octet+8]
            bst_ip_octets.append(bst_ip_octet)
        bst_ip_address = []
        for each_octet in bst_ip_octets:
            bst_ip_address.append(str(int(each_octet, 2)))

        broadcast_address = ".".join(bst_ip_address)
        print ("\n")
        print ("Network address is: %s" % network_address)
        print ("Broadcast address is: %s" % broadcast_address)
        print ("Number of valid hosts per subnet: %s" % no_of_hosts)
        print ("Wildcard mask: %s" % wildcard_mask)
        print ("Mask bits: %s" % no_of_ones)
        print ("\n")
        while True:
            generate = input("Generate random ip address from subnet? (y/n)")
            if generate == "y":
                generated_ip = []
                for indexb, oct_bst in enumerate(bst_ip_address):
```



```python
                    for indexn, oct_net in enumerate(net_ip_address):
                        if indexb == indexn:
                            if oct_bst == oct_net:
                                generated_ip.append(oct_bst)
                            else:
                               generated_ip.append(str(random.randint(int(oct_net),
int(oct_bst))))
                y_iaddr = ".".join(generated_ip)
                print("Random IP address is: %s" % y_iaddr)
                print ("\n")
                continue
            else:
                print ("Ok, not possible!\n")
                break
    except KeyboardInterrupt:
        print ("\n\nProgram aborted by user. Exiting...\n")
        sys.exit()
subnet_calc()
```

OUTPUT:



```
Enter a subnet mask: 255.255.255.0


Network address is: 192.168.10.0
Broadcast address is: 192.168.10.255
Number of valid hosts per subnet: 254
Wildcard mask: 0.0.0.255
Mask bits: 24


Generate random ip address from subnet? (y/n)Y
Ok, not possible!

PS X:\WORKSPACE> Y
```