DAA CODE

RANDOMIZED QUICH SHORT:

```
import random
def Ran_quick_sort(arr):
    n=len(arr)
    if n<=1:
        return arr
    pivot = random.choice(arr)
    left = []
    right = []
    pivot_List=[]
    for i in arr:
        if(i < pivot):
            left.append(i)
        elif i<pivot:
            right.append(i)
        else:
            pivot_List.append(i)
    return Ran_quick_sort(left) +pivot_List+ Ran_quick_sort(right)



n=int(input("Enter the number of elements in array:"))
arr=[]
for i in range(0,n):
    a=int(input("Enter the element or array:"))
    arr.append(a)
print("The original Array or List:\n",arr)
r=Ran_quick_sort(arr)
print("Arral after shorted",r)
```

RSA:

```
import random
def gcd(a,b):
    while b!=0:
        a=b
        b=a%b
    return a
def is_prime(num):
    if num>1:
        for i in range(2,(num**(1//2))+1):
            if(num% i==0):
                return False
    else:
        return False
    return True
```

```python
def pr_gen():
    while True:
        num=random.randint(2,100)
        if is_prime(num):
            return num

def gcd(n1,n2):
    while(n2!=0):
        n1,n2=n2,n1%n2
    return n1



def co_prim(a):
    while True:
        num=random.randint(2,a)
        if gcd(a,num)==1:
            return num

def invers_mod(a,de):
    while True:
        # num=random.randint(2,a)
        for i in range(1,de):
            if(a*i)%de==1:
                return i
        return None

def str_dec(msg):
    int_m=[]
    print("string to decimal")
    for i in msg:
        int_m.append(ord(i))
    print(int_m)
    return int_m

def enc(msg,e,n):
    en=str_dec(msg)
    new_msg=[]
    for i in en:
        new_msg.append((i**e)%n)
    # en=[chr(i) for i in en]
    return new_msg

def dec(new_msg,d,n):
    omsg=[]
    for i in new_msg:
        omsg.append((i**d)%n)
    print("new string list is:",omsg)
    return ''.join(chr(i) for i in omsg)


msg=input("enter the input string:\n")
print("this is original message:", msg)
p=pr_gen()
q=pr_gen()
while p==q:
    q=pr_gen()
n=p*q
del_n = (p-1)*(q-1)
e = co_prim(del_n)
d = invers_mod(e,del_n)

new_msg=[]
new_msg=enc(msg,e,n)
print("encripted Message",new_msg)
or_msg=[]
or_msg=dec(new_msg,d,n)
# print(or_msg)
print(msg)
```

BAYER-MOOR ALGORITHM:

RSA:

```
import random
def gcd(a,b):
    while b!=0:
        a=b
        b=a%b
    return a
def is_prime(num):
    if num>1:
        for i in range(2,(num**(1//2))+1):
            if(num% i==0):
                return False
    else:
        return False
    return True

def pr_gen():
    while True:
        num=random.randint(2,100)
        if is_prime(num):
            return num
```

```python
def gcd(n1,n2):
    while(n2!=0):
        n1,n2=n2,n1%n2
    return n1



def co_prim(a):
    while True:
        num=random.randint(2,a)
        if gcd(a,num)==1:
            return num

def invers_mod(a,de):
    while True:
        # num=random.randint(2,a)
        for i in range(1,de):
            if(a*i)%de==1:
                return i
        return None

def str_dec(msg):
    int_m=[]
    print("string to decimal")
    for i in msg:
        int_m.append(ord(i))
    print(int_m)
    return int_m

def enc(msg,e,n):
    en=str_dec(msg)
    new_msg=[]
    for i in en:
        new_msg.append((i**e)%n)
    # en=[chr(i) for i in en]
    return new_msg

def dec(new_msg,d,n):
    omsg=[]
    for i in new_msg:
        omsg.append((i**d)%n)
    print("new string list is:",omsg)
    return ''.join(chr(i) for i in omsg)


msg=input("enter the input string:\n")
print("this is original message:", msg)
p=pr_gen()
q=pr_gen()
```

```python
while p==q:
    q=pr_gen()
n=p*q
del_n = (p-1)*(q-1)
e = co_prim(del_n)
d = invers_mod(e,del_n)

new_msg=[]
new_msg=enc(msg,e,n)
print("encripted Message",new_msg)
or_msg=[]
or_msg=dec(new_msg,d,n)
# print(or_msg)
print(msg)
```