#circular Queue implemntation with fixed size

1.initialize: -size -Rear pointer -Qeue (list) 2.Enqueue: -Queue is full(rear+1 == front) -first element(rear = front = 0),then append -Entering anywhere except at the begining (rear = rear+1%size),then append

```
class circularQ:
    def __init__(self,size):
        self.size = size
        self.front = self.rear=-1
        self.queue = [None]*size
    def enqueue(self,data):
        if(self.rear+1)%self.size == self.front:
            print("Queue is Full")
        elif self.rear ==-1:
            self.front = self.rear=0
            self.queue[self.rear] = data
        else:
            self.rear = (self.rear+1)%self.size
            self.queue[self.rear]=data
    def dequeue(self):
        if self.front==-1:
            popped = self.queue[self.front]
            print("Q is empty")
            return
        elif self.front==self.rear:
            self.front=self.rear=-1
            return popped
        else:
            self.front=(self.front+1)%self.size
            return popped
    def display(self):
        if self.front == -1:
            print("Queue  is empty")
            return
        i=self.front
        while True:
            print(self.queue[i])
            if i == self.rear:
                break
            i=(i+1)%self.size




j = circularQ(3)
j.enqueue(90)
```

```
j.enqueue(80)
j.enqueue(50)

j.enqueue(100)
```

```
Queue is Full
```

```
j = circularQ(3)
j.dequeue()
j.dequeue()
j.dequeue()
```

```
Q is empty
Q is empty
Q is empty
```

```
k=circularQ(3)
k.enqueue(80)
k.enqueue(70)
k.enqueue(90)
k.display()
```

```
80
70
90
```

```
def a(m,n):
    q=0
    p=0
    i=1
    for i in range(m+1):
        if i%n == 0:
            p=p+i
        else:
            q=q+i

    c=q-p
    print(c)

a(20,4)
```

```
90
```

```
def checkpassword(char,n):
    n = len(char)
    if n>4:
        return 1
    else:
        return 0
```