DLL:

```
def createNode(data):
    return {"data": data, "next": None, "prev": None}
def insertAtHead(DLL, data):
    newNode = createNode(data)
    if DLL is None:
        return newNode
    newNode["next"] = DLL
    DLL["prev"] = newNode
    return newNode
def insertAtTail(DLL, data):
    newNode = createNode(data)
    if DLL is None:
        return newNode
    current = DLL
    while current["next"] is not None:
        current = current["next"]
    current["next"] = newNode
    newNode["prev"] = current
    return DLL
def printAtHead(DLL):
    current = DLL
    while current["next"] is not None:
        print(current["data"], end="->")
        current = current["next"]
    print(current["data"])
def printAtTail(DLL):
    current = DLL
    while current["next"] is not None:
        current = current["next"]
    while current["prev"] is not None:
        print(current["data"], end="<-")
        current = current["prev"]
    print(current["data"])
def printRange(DLL, start, end):
    current = DLL
    for i in range(start):
        current = current["next"]
    for i in range(start, end):
        print(current["data"], end="->")
        current = current["next"]
    print(current["data"])
```

Queue:

```
def enqueue(queue, queueMaxCapacity, newData):
    frontIndex = queueMaxCapacity - 1
    rearIndex = 0
    if queue[frontIndex] is None:
        queue[frontIndex] = newData
        return queue
    if queue[rearIndex] is not None:
        print('antrian penuh')
```

```
        return queue
    for i in
range(queueMaxCapacity-2, -1, -1):
        if queue[i] is None:
            queue[i] = newData
            return queue
def queueIsEmpty(frontIndex,
queue):
    return queue[frontIndex] is
None
def dequeue(queue,
queueMaxCapacity):
    frontIndex = -1
    tempQueue = queue
    queue = [None] *
queueMaxCapacity
    if queueIsEmpty(frontIndex,
tempQueue):
        return tempQueue
    for i in
range(queueMaxCapacity-2, -1, -1):
        queue[i+1] = tempQueue[i]
    return queue
def countAvailable(queue,
queueMaxCapacity):
    count = 0
    for i in
range(queueMaxCapacity):
        if queue[i] is None:
            count += 1
    return count
def countData(queue,
queueMaxCapacity):
    count = 0
    for i in
range(queueMaxCapacity):
        if queue[i] is not None:
            count += 1
    return count
Stack:
def
calculate_sum_of_squares(stack):
    total = 0
    for i in range(len(stack)):
        total += stack[i] *
stack[i]
    return total
def
calculate_product_of_squares(stack)
:
    total = 1
    for i in range(len(stack)):
        total *= stack[i] *
stack[i]
    return total
```