**arrays**

class StackArray:

def\_int\_(self):

self.items = []

def is\_empty(self)

return len(self.items) == 0

def push(self,item):

self.items.append(item)

def pop(self):

if not self.is\_empty():

return self.items.pop()

else:

raise IndexError("Pop from an empty stack")

def peek(self):

if not self.is\_empty():

return self.items[-1]

else:

raise IndexError("Peek from an empty stack")

def size(self):

return len(self.items)

class QueueArray:

def\_int\_(self):

self.items = []

def is\_empty(self):

return len(self.item) == 0

def enqueue(self,item):

self.items.append(item)

def dequeue(self):

if not self.is\_empty():

return self.items.pop(0)

else:

raise IndexError("Dequeue from an empty queue")

def front(self):

if not self.is\_empty():

return self.item[0]

else:

raise IndexError("Front from an empty queue")

def size(self):

return len(self.items)

class Node:

def\_int\_(self,data):

self.data = data

self.next = None

class QueueLinkList:

def\_int\_(self):

self.front = self.rear = None

def is\_empty(self):

return self.front is None

def enqueue(self,item):

new\_node = Node(item)

if self.is\_empty():

self.front = self.rear = new\_node

else:

self.rear.next = new\_node

self.rear = new\_node

def dequeue(self):

if not self.is\_empty():

dequeue = self.front.data

self.front = self.front.next

if self.front is None:

self.rear = None

return dequeue

else:

raise IndexError("Dequeue from an empty queue")

def front(self):

if not self.is\_empty():

return self.front.data

else:

raise IndexError("Front from an empty queue")