```
def enqueue(self, item):
    if self.num >= self.max_size:
        raise Exception("Queue overflow
    self.Q[(self.num + self.first) % se
    self.num += 1

def dequeue(self):
    if self.num == 0:
        raise Exception("Queue empty")
    item = self.Q[self.first]
    self.first = (self.first + 1) % sel
    self.num -= 1
    return item

def front(self):
    if self.num == 0:
        raise Exception("Queue empty")
    return self.Q[self.first]

def is_empty(self):
    return self.num == 0

def size(self):
    return self.num

def is_full(self):
    return self.num >= self.max_size

# Add this method to output the i-th el
def get(self, i):
    if i < 0 or i >= self.num:
        raise Exception("Index out of r
    return self.Q[(self.first + i) % se
```

```
def pyramid_sort(arr):
    stack = [ ]
    max_value = None

    while len(stack) > 0:
        max_value = max(stack,
key=stack.pop)
        for i in range(len(arr)):
            if arr[i] > max_value:
                stack.append(max_value)
                max_value = arr[i]
                break

    return arr
```

```
def delete(self,x):
        if x <=self.num:
            item1=self.Q[x]
            item2=x+1
            self.Q=self.Q[:x]
+self.Q[item2: ]
        else:
            raise Exception("error")
        return self.Q,item1
```