Infix to Postfix:

```
Operators = set(['+', '-', '*', '/', '(', ')', '^'])

Priority = {'+':1, '-':1, '*':2, '/':2, '^':3}

def infixToPostfix(expression):

    stack = []

    output = ''


    for character in expression:

        if character not in Operators:

            output+= character

        elif character=='(':

            stack.append('(')

        elif character==')':

            while stack and stack[-1]!= '(':

                output+=stack.pop()

            stack.pop()

        else:

            while stack and stack[-1]!='(' and
Priority[character]<=Priority[stack[-1]]:

                output+=stack.pop()

            stack.append(character)

    while stack:

        output+=stack.pop()

    return output


expression = input('Enter infix expression ')

print('infix notation: ',expression)

print('postfix notation: ',infixToPostfix(expression))
```

Queue:

```
class Queue:
    def __init__(self):
        self.items = []

    def is_empty(self):
        return len(self.items) == 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("Cannot dequeue from an empty queue")

    def size(self):
        return len(self.items)

queue = Queue()

queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)

print(queue.dequeue())
print(queue.dequeue())

print(queue.is_empty())


print(queue.size())
```