```python
#!/usr/bin/env python3

# QUESTION 1 ~ Worked on by Jade Hudson and Sruthi Santhosh
class Queue:
    def __init__(self):
        self.items = []

    def is_empty(self):
        return self.items == []

    def enqueue(self, item):
        self.items.insert(0, item)

    def dequeue(self):
        return self.items.pop()

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

    def reverse(self):
        return self.items[::-1]

    def put(self, item):
        self.items.append(item)

    def get(self):
        if len(self.items) < 1:
            return None
        return self.items.pop(0)

    def front(self):
        return self.items[0]


def Sort_Recursion3(q, qsize) :
    if qsize <= 0:
        return
    q.put(q.get())
    Sort_Recursion3(q, qsize - 1)


def Sort_Recursion2(q, tmp, qsize) :
    if q.is_empty() or qsize == 0:
        q.put(tmp)  # adds tmp to end of queue
        return
    elif tmp <= q.front():
        q.put(tmp)
        Sort_Recursion3(q, qsize)
    else :
        q.put(q.get())  # gets and removes the first in queue and places at end of queue
        Sort_Recursion2(q, tmp, qsize - 1)


def Sort_Recursion1(q):
    if q.is_empty():
        return
    tmp = q.get()  # gets and removes first in queue
    Sort_Recursion1(q)
    Sort_Recursion2(q, tmp, q.size())


def Recursive_Print(sorted_q):
    return q.items


def Reverse_N(q, n):
    return q[:n]


if __name__ == '__main__':
    q = Queue()
    q.enqueue(1)
    q.enqueue(2)
    q.enqueue(3)
    print(q.dequeue())
    print(q.size())
    print(q.dequeue())
    print(q.dequeue())
    print(q.is_empty())
    print(q.size())

    q.enqueue(1)
    q.enqueue(9)
    q.enqueue(5)
    q.enqueue(3)
    print(q.items)
    print(q.reverse())
    print(Recursive_Print(Sort_Recursion1(q)))
    print(Reverse_N(q.reverse(), 2))



# QUESTION 2 ~ Worked on by Jade Hudson and Sruthi Santhosh

def Convert_Binary(n):
    q = Queue()  # use queue from previous question
    q.put(1)
    while(n > 0):
        n = n - 1
        start = q.get()
        print(start)
        end = start
        q.put(str(start)+"0")
        q.put(str(end)+"1")

if __name__ == '__main__':
    n = 16
    Convert_Binary(n)



# QUESTION 3 ~ Worked on by Jade Hudson and Sruthi Santhosh

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

    def is_empty(self):
        return self.items == []
        # return len(self.items) == 0

    def push(self, item):
        self.items.append(item)

    def pop(self):
        return self.items.pop()


stack1 = Stack()
s = 'EAS*Y*QUE***ST***IO*N***'
i = 0
result = []
while i < len(s):
    if s[i].isalpha():
        stack1.push(s[i])
    else:
        result.append(stack1.pop())
    i = i + 1
print(result)



# QUESTION 3 # Worked on by Jade Hudson and Sruthi Santhosh
class Queue:
    def __init__(self):
        self.items = []

    def is_empty(self):
        return self.items == []

    def enqueue(self, item):
        self.items.insert(0, item)

    def dequeue(self):
        return self.items.pop()

    def put(self, item):
        self.items.append(item)

    def get(self):
        if len(self.items) < 1:
            return None
        return self.items.pop(0)

queue1 = Queue()
s = 'EAS*Y*QUE***ST***IO*N***'
i = 0
result = []
while i < len(s):
    if s[i].isalpha():
        queue1.put(s[i])
    else:
        result.append(queue1.get())
    i = i + 1
print(result)



# QUESTION 5 ~ Worked on by Jade Hudson and Sruthi Santhosh
class Stack:
    def __init__(self):
        self.items = []

    def is_empty(self):
        return self.items == []
        # return len(self.items) == 0

    def push(self, item):
        self.items.append(item)

    def pop(self):
        return self.items.pop()

    def reverse(self):
        return self.items[::-1]

s = "abcdefg"
stack1 = Stack()
i = 0
while i < len(s):
    stack1.push(s[i])
    i = i + 1
print(stack1.reverse())



# QUESTION 6 ~ Worked on by Jade Hudson and Sruthi Santhosh
class Stack:
    def __init__(self):
        self.items = []

    def is_empty(self):
        return self.items == []
        # return len(self.items) == 0

    def push(self, item):
        self.items.append(item)

    def pop(self):
        return self.items.pop()

    def reverse(self):
        return self.items[::-1]

def postfix(stack1, s):
    for val in s:
        if val.isdigit():
            stack1.push(val)
        else:
            num1 = int(stack1.pop())
            num2 = int(stack1.pop())
            if val == "/":
                stack1.push(num2 / num1)
            else:
                switcher ={'+':int(num2) + int(num1), '-':int(num2)-int(num1), '*':int(num2) * int(num1), '^':int(num2)**int(num1)}
                stack1.push(switcher.get(val))
    return int(stack1.pop())

stack1 = Stack()
s = "1432^*+147--+"
print(postfix(stack1, s))
```