Implement Queue using Linked listsd

```
class Node:
    def _init_(self, value=None):
        self.value = value
        self.next = None

class LinkedListQueue:
    def _init_(self):
        self.front = None  # Points to the front of the queue
        self.rear = None   # Points to the rear of the queue
        self.size = 0      # Keeps track of the number of elements

    def enqueue(self, value):
        new_node = Node(value)
        if self.rear is None:  # The queue is empty
            self.front = new_node
            self.rear = new_node
        else:
            self.rear.next = new_node
            self.rear = new_node
        self.size += 1

    def dequeue(self):
        if self.front is None:  # The q
            raise IndexError("Dequeue from an empty queue")
        value = self.front.value
        self.front = self.front.next
        if self.front is None:  # The queue is now empty
            self.rear = None
        self.size -= 1
        return value

    def peek(self):
        if self.front is None:  # The queue is empty
            raise IndexError("Peek from an empty queue")
        return self.front.value

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

    def get_size(self):
        return self.size

    def _str_(self):
        values = []
        current = self.front
        while current:
            values.append(current.value)
```

```
            current = current.next
        return " -> ".join(map(str, values))


if __name_ == "__main__":
    queue = LinkedListQueue()
    queue.enqueue(1)
    queue.enqueue(2)
    queue.enqueue(3)

print(queue)
print(queue.dequeue())
print(queue.peek())
print(queue)
print(queue.get_size())
```

2

Write a function to reverse a queue

```
from collections import deque

def reverse_queue(queue):

    stack = []



    while queue:
        stack.append(queue.popleft())


    while stack:
        queue.append(stack.pop())

if __name__ == "__main__":
    q = deque([1, 2, 3, 4, 5])
    print("Original queue:", list(q))
    reverse_queue(q)
    print("Reversed queue:", list(q))
```

```
Original queue: [1, 2, 3, 4, 5]
Reversed queue: [5, 4, 3, 2, 1]
```

1. You are given a function. CheckPassword(char, n); The function accepts string str of size n as an argument. Implement the function which returns 1 if given string str is valid password else 0. str is a valid password if it satisfies the below conditions. ● At least 4 characters ● At least one numeric digit ● At Least one Capital Letter ● Must not have space or slash (/) ● Starting character must not be a number

```
def checkPassword(str):
    valid = False
    if len(str)<4:
        return 0
    for i in str:
        if i.isdigit():
            valid = True
            break
    if valid==False:
        return 0
    valid = False
    for i in str:
        if i.isupper():
            valid = True
            break
    if valid==False:
        return 0
    for i in str:
        if i==" " or i=="/":
            return 0
    if str[0].isdigit():
        return 0
    return 1

pwd = input("Enter a password to check: ")
print(checkPassword(pwd))
```

```
Enter a password to check:  gfdgsdjbkg6763

0
```