```python
# 1 . Queue using Two Stacks (Array or List-based):
class QueueUsingStacks:
    def __init__(self):
        self.stack1 = []
        self.stack2 = []

    def is_empty(self):
        return not self.stack1 and not self.stack2

    def enqueue(self, x):
        self.stack1.append(x)

    def dequeue(self):
        if self.is_empty():
            print("Queue Underflow")
            return None
        if not self.stack2:
            while self.stack1:
                self.stack2.append(self.stack1.pop())
        return self.stack2.pop()

    def peek(self):
        if self.is_empty():
            print("Queue Underflow")
            return None
        if not self.stack2:
            while self.stack1:
                self.stack2.append(self.stack1.pop())
        return self.stack2[-1]

    def display(self):
        temp = self.stack2[::-1] + self.stack1
        print("Queue:", temp)

q = QueueUsingStacks()
while True:
    print("\n1. Enqueue\n2. Dequeue\n3. Peek\n4. Display\n5. Exit")
    ch = int(input("Enter your choice: "))
    if ch == 1:
        num = int(input("Enter data: "))
        q.enqueue(num)
    elif ch == 2:
        val = q.dequeue()
        if val is not None:
            print("Dequeued:", val)
    elif ch == 3:
        val = q.peek()
        if val is not None:
            print("Front item:", val)
    elif ch == 4:
```

```
        q.display()
    elif ch == 5:
        print("Quitting...")
        break
    else:
        print("Invalid choice")


1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit

Enter your choice:  1
Enter data:  23


1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit

Enter your choice:  1
Enter data:  34


1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit

Enter your choice:  2

Dequeued: 23

1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit

Enter your choice:  1
Enter data:  44


1. Enqueue
2. Dequeue
3. Peek
```

```
4. Display
5. Exit

Enter your choice:  3

Front item: 34

1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit

Enter your choice:  4

Queue: [34, 44]

1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Exit

Enter your choice:  5

Quitting...
```

```python
# 2 . Reverse a Queue Using Recursion:
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedQueue:
    def __init__(self):
        self.front = None
        self.rear = None

    def isempty(self):
        return self.front is None

    def enqueue(self, data):
        newNode = Node(data)
        if self.rear is None:
            self.front = self.rear = newNode
        else:
            self.rear.next = newNode
            self.rear = newNode

    def dequeue(self):
        if self.isempty():
```

```
            print("Queue Underflow")
            return None
        data = self.front.data
        self.front = self.front.next
        if self.front is None:
            self.rear = None
        return data

    def peek(self):
        if self.isempty():
            print("Queue Underflow")
            return None
        return self.front.data

    def display(self):
        if self.isempty():
            print("Queue is empty")
            return
        temp = self.front
        while temp:
            print(temp.data, end=" -> ")
            temp = temp.next
        print("None")

    def reverse(self):
        if self.isempty():
            return
        data = self.dequeue()
        self.reverse()
        self.enqueue(data)

# Menu-driven program
q = LinkedQueue()

while True:
    print("\nProgram to reverse a queue using recursion")
    print("1. Enqueue")
    print("2. Dequeue")
    print("3. Peek")
    print("4. Display")
    print("5. Reverse Queue")
    print("6. Exit")
    choice = int(input("Enter your choice: "))

    if choice == 1:
        num = int(input("Enter the data: "))
        q.enqueue(num)
    elif choice == 2:
        val = q.dequeue()
        if val is not None:
```

```
            print("Dequeued:", val)
    elif choice == 3:
        val = q.peek()
        if val is not None:
            print("Front item:", val)
    elif choice == 4:
        q.display()
    elif choice == 5:
        q.reverse()
        print("Queue reversed.")
    elif choice == 6:
        print("Exiting...")
        break
    else:
        print("Invalid choice, try again.")


Program to reverse a queue using recursion
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Reverse Queue
6. Exit

Enter your choice:  1
Enter the data:  22


Program to reverse a queue using recursion
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Reverse Queue
6. Exit

Enter your choice:  1
Enter the data:  33


Program to reverse a queue using recursion
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Reverse Queue
6. Exit

Enter your choice:  1
Enter the data:  44
```

```
Program to reverse a queue using recursion
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Reverse Queue
6. Exit

Enter your choice:  1
Enter the data:  55


Program to reverse a queue using recursion
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Reverse Queue
6. Exit

Enter your choice:  2

Dequeued: 22

Program to reverse a queue using recursion
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Reverse Queue
6. Exit

Enter your choice:  3

Front item: 33

Program to reverse a queue using recursion
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Reverse Queue
6. Exit

Enter your choice:  4

33 -> 44 -> 55 -> None

Program to reverse a queue using recursion
1. Enqueue
2. Dequeue
```

```
3. Peek
4. Display
5. Reverse Queue
6. Exit

Enter your choice:  5

Queue reversed.

Program to reverse a queue using recursion
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Reverse Queue
6. Exit

Enter your choice:  4

55 -> 44 -> 33 -> None

Program to reverse a queue using recursion
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Reverse Queue
6. Exit

Enter your choice:  6

Exiting...
```

```python
# 3. Design a Queue that Supports max() Operation:
class MaxQueue:
    def __init__(self):
        self.queue = []
        self.max_queue = []  # stores potential max elements

    def isempty(self):
        return len(self.queue) == 0

    def enqueue(self, x):
        self.queue.append(x)
        while self.max_queue and self.max_queue[-1] < x:
            self.max_queue.pop()
        self.max_queue.append(x)

    def dequeue(self):
        if self.isempty():
            print("Queue Underflow")
```

```
            return None
        val = self.queue.pop(0)
        if val == self.max_queue[0]:
            self.max_queue.pop(0)
        return val

    def peek(self):
        if self.isempty():
            print("Queue Underflow")
            return None
        return self.queue[0]

    def get_max(self):
        if self.isempty():
            print("Queue is empty, no max")
            return None
        return self.max_queue[0]

    def display(self):
        if self.isempty():
            print("Queue is empty")
        else:
            print("Queue:", self.queue)

# Menu-driven program
q = MaxQueue()

while True:
    print("\nQueue with max() operation")
    print("1. Enqueue")
    print("2. Dequeue")
    print("3. Peek")
    print("4. Display")
    print("5. Get Max")
    print("6. Exit")
    choice = int(input("Enter your choice: "))

    if choice == 1:
        num = int(input("Enter data: "))
        q.enqueue(num)
    elif choice == 2:
        val = q.dequeue()
        if val is not None:
            print("Dequeued:", val)
    elif choice == 3:
        val = q.peek()
        if val is not None:
            print("Front item:", val)
    elif choice == 4:
        q.display()
```

```
    elif choice == 5:
        max_val = q.get_max()
        if max_val is not None:
            print("Max element:", max_val)
    elif choice == 6:
        print("Exiting...")
        break
    else:
        print("Invalid choice, try again.")


Queue with max() operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Get Max
6. Exit

Enter your choice:  1
Enter data:  11


Queue with max() operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Get Max
6. Exit

Enter your choice:  22

Invalid choice, try again.

Queue with max() operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Get Max
6. Exit

Enter your choice:  1
Enter data:  22


Queue with max() operation
1. Enqueue
2. Dequeue
3. Peek
```

```
4. Display
5. Get Max
6. Exit

Enter your choice:  1
Enter data:  33


Queue with max() operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Get Max
6. Exit

Enter your choice:  1
Enter data:  44


Queue with max() operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Get Max
6. Exit

Enter your choice:  2

Dequeued: 11

Queue with max() operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Get Max
6. Exit

Enter your choice:  3

Front item: 22

Queue with max() operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Get Max
6. Exit
```

```
Enter your choice:  4

Queue: [22, 33, 44]

Queue with max() operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Get Max
6. Exit

Enter your choice:  5

Max element: 44

Queue with max() operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Get Max
6. Exit

Enter your choice:  6

Exiting...
```

```python
# 4. Merge Two Queues:
class Queue:
    def __init__(self):
        self.items = []

    def enqueue(self, data):
        self.items.append(data)

    def dequeue(self):
        if self.isempty():
            return None
        return self.items.pop(0)

    def isempty(self):
        return len(self.items) == 0

    def display(self):
        if self.isempty():
            print("Queue is empty")
        else:
            print("Queue:", self.items)

def merge_queues(q1, q2):
```

```
    merged = Queue()
    while not q1.isempty() or not q2.isempty():
        if not q1.isempty():
            merged.enqueue(q1.dequeue())
        if not q2.isempty():
            merged.enqueue(q2.dequeue())
    return merged

# Menu-driven for two queues
q1 = Queue()
q2 = Queue()
merged = None

while True:
    print("\nMerge Two Queues Alternately")
    print("1. Enqueue to Queue 1")
    print("2. Enqueue to Queue 2")
    print("3. Display Queue 1")
    print("4. Display Queue 2")
    print("5. Merge Queues")
    print("6. Display Merged Queue")
    print("7. Exit")

    choice = int(input("Enter your choice: "))

    if choice == 1:
        num = int(input("Enter data for Queue 1: "))
        q1.enqueue(num)
    elif choice == 2:
        num = int(input("Enter data for Queue 2: "))
        q2.enqueue(num)
    elif choice == 3:
        print("Queue 1:")
        q1.display()
    elif choice == 4:
        print("Queue 2:")
        q2.display()
    elif choice == 5:
        merged = merge_queues(q1, q2)
        print("Queues merged successfully.")
    elif choice == 6:
        if merged:
            print("Merged Queue:")
            merged.display()
        else:
            print("You need to merge the queues first.")
    elif choice == 7:
        print("Exiting...")
        break
```

```
    else:
        print("Invalid choice, try again.")


Merge Two Queues Alternately
1. Enqueue to Queue 1
2. Enqueue to Queue 2
3. Display Queue 1
4. Display Queue 2
5. Merge Queues
6. Display Merged Queue
7. Exit

Enter your choice:  1
Enter data for Queue 1:  11


Merge Two Queues Alternately
1. Enqueue to Queue 1
2. Enqueue to Queue 2
3. Display Queue 1
4. Display Queue 2
5. Merge Queues
6. Display Merged Queue
7. Exit

Enter your choice:  1
Enter data for Queue 1:  22


Merge Two Queues Alternately
1. Enqueue to Queue 1
2. Enqueue to Queue 2
3. Display Queue 1
4. Display Queue 2
5. Merge Queues
6. Display Merged Queue
7. Exit

Enter your choice:  1
Enter data for Queue 1:  33


Merge Two Queues Alternately
1. Enqueue to Queue 1
2. Enqueue to Queue 2
3. Display Queue 1
4. Display Queue 2
5. Merge Queues
6. Display Merged Queue
7. Exit
```

```
Enter your choice:  2
Enter data for Queue 2:  99


Merge Two Queues Alternately
1. Enqueue to Queue 1
2. Enqueue to Queue 2
3. Display Queue 1
4. Display Queue 2
5. Merge Queues
6. Display Merged Queue
7. Exit

Enter your choice:  2
Enter data for Queue 2:  88


Merge Two Queues Alternately
1. Enqueue to Queue 1
2. Enqueue to Queue 2
3. Display Queue 1
4. Display Queue 2
5. Merge Queues
6. Display Merged Queue
7. Exit

Enter your choice:  2
Enter data for Queue 2:  77


Merge Two Queues Alternately
1. Enqueue to Queue 1
2. Enqueue to Queue 2
3. Display Queue 1
4. Display Queue 2
5. Merge Queues
6. Display Merged Queue
7. Exit

Enter your choice:  3

Queue 1:
Queue: [11, 22, 33]

Merge Two Queues Alternately
1. Enqueue to Queue 1
2. Enqueue to Queue 2
3. Display Queue 1
4. Display Queue 2
5. Merge Queues
```

```
6. Display Merged Queue
7. Exit

Enter your choice:  4

Queue 2:
Queue: [99, 88, 77]

Merge Two Queues Alternately
1. Enqueue to Queue 1
2. Enqueue to Queue 2
3. Display Queue 1
4. Display Queue 2
5. Merge Queues
6. Display Merged Queue
7. Exit

Enter your choice:  5

Queues merged successfully.

Merge Two Queues Alternately
1. Enqueue to Queue 1
2. Enqueue to Queue 2
3. Display Queue 1
4. Display Queue 2
5. Merge Queues
6. Display Merged Queue
7. Exit

Enter your choice:  6

Merged Queue:
Queue: [11, 99, 22, 88, 33, 77]

Merge Two Queues Alternately
1. Enqueue to Queue 1
2. Enqueue to Queue 2
3. Display Queue 1
4. Display Queue 2
5. Merge Queues
6. Display Merged Queue
7. Exit

Enter your choice:  7

Exiting...
```

```python
# 5. Implement a Queue with Count of Specific Element:
class WordQueue:
    def __init__(self):
        self.queue = []
```

```
    def enqueue(self, word):
        self.queue.append(word)

    def dequeue(self):
        if self.isempty():
            return None
        return self.queue.pop(0)

    def isempty(self):
        return len(self.queue) == 0

    def display(self):
        print("Queue (Words):", self.queue)

    def reverse(self):
        self.queue.reverse()

    def get_sentence(self):
        return ' '.join(self.queue)

# Menu-driven program
q = WordQueue()

while True:
    print("\nProgram to reverse words in a sentence using a queue")
    print("1. Input a sentence")
    print("2. Display queue")
    print("3. Reverse words")
    print("4. Display reversed sentence")
    print("5. Exit")

    choice = int(input("Enter your choice: "))

    if choice == 1:
        sentence = input("Enter a sentence: ")
        q = WordQueue()  # reset queue
        for word in sentence.strip().split():
            q.enqueue(word)
        print("Words enqueued.")
    elif choice == 2:
        q.display()
    elif choice == 3:
        q.reverse()
        print("Words reversed.")
    elif choice == 4:
        print("Reversed sentence:", q.get_sentence())
    elif choice == 5:
        print("Exiting...")
        break
```

```
    else:
        print("Invalid choice, try again.")


Program to reverse words in a sentence using a queue
1. Input a sentence
2. Display queue
3. Reverse words
4. Display reversed sentence
5. Exit

Enter your choice:  1
Enter a sentence:  I love my self

Words enqueued.

Program to reverse words in a sentence using a queue
1. Input a sentence
2. Display queue
3. Reverse words
4. Display reversed sentence
5. Exit

Enter your choice:  2

Queue (Words): ['I', 'love', 'my', 'self']

Program to reverse words in a sentence using a queue
1. Input a sentence
2. Display queue
3. Reverse words
4. Display reversed sentence
5. Exit

Enter your choice:  3

Words reversed.

Program to reverse words in a sentence using a queue
1. Input a sentence
2. Display queue
3. Reverse words
4. Display reversed sentence
5. Exit

Enter your choice:  4

Reversed sentence: self my love I

Program to reverse words in a sentence using a queue
1. Input a sentence
2. Display queue
```

```
3. Reverse words
4. Display reversed sentence
5. Exit

Enter your choice:  5

Exiting...
```

```python
# 6. Implement a Queue to Reverse Words in a Sentence:
class CustomQueue:
    def __init__(self):
        self.queue = []

    def enqueue(self, data):
        self.queue.append(data)

    def dequeue(self):
        if self.isempty():
            print("Queue Underflow")
            return None
        return self.queue.pop(0)

    def isempty(self):
        return len(self.queue) == 0

    def peek(self):
        if self.isempty():
            print("Queue Underflow")
            return None
        return self.queue[0]

    def display(self):
        if self.isempty():
            print("Queue is empty")
        else:
            print("Queue:", self.queue)

    def contains(self, x):
        return x in self.queue

# Menu-driven program
q = CustomQueue()

while True:
    print("\nQueue with contains(x) operation")
    print("1. Enqueue")
    print("2. Dequeue")
    print("3. Peek")
    print("4. Display")
    print("5. Check if element exists (contains)")
    print("6. Exit")
```

```
    choice = int(input("Enter your choice: "))

    if choice == 1:
        num = int(input("Enter data: "))
        q.enqueue(num)
    elif choice == 2:
        val = q.dequeue()
        if val is not None:
            print("Dequeued:", val)
    elif choice == 3:
        val = q.peek()
        if val is not None:
            print("Front item:", val)
    elif choice == 4:
        q.display()
    elif choice == 5:
        x = int(input("Enter element to check: "))
        if q.contains(x):
            print(f"{x} exists in the queue.")
        else:
            print(f"{x} not found in the queue.")
    elif choice == 6:
        print("Exiting...")
        break
    else:
        print("Invalid choice, try again.")


Queue with contains(x) operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if element exists (contains)
6. Exit

Enter your choice:  1
Enter data:  22


Queue with contains(x) operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if element exists (contains)
6. Exit

Enter your choice:  1
Enter data:  33
```

```
Queue with contains(x) operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if element exists (contains)
6. Exit

Enter your choice:  1
Enter data:  44


Queue with contains(x) operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if element exists (contains)
6. Exit

Enter your choice:  2

Dequeued: 22

Queue with contains(x) operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if element exists (contains)
6. Exit

Enter your choice:  3

Front item: 33

Queue with contains(x) operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if element exists (contains)
6. Exit

Enter your choice:  4

Queue: [33, 44]

Queue with contains(x) operation
1. Enqueue
2. Dequeue
```

```
3. Peek
4. Display
5. Check if element exists (contains)
6. Exit

Enter your choice:  5
Enter element to check:  44

44 exists in the queue.

Queue with contains(x) operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if element exists (contains)
6. Exit

Enter your choice:  6

Exiting...
```

```python
# 7. Implement a Queue that Supports contains(x) Operation:
class CustomQueue:
    def __init__(self):
        self.queue = []

    def enqueue(self, data):
        self.queue.append(data)

    def dequeue(self):
        if self.isempty():
            print("Queue Underflow")
            return None
        return self.queue.pop(0)

    def isempty(self):
        return len(self.queue) == 0

    def peek(self):
        if self.isempty():
            print("Queue Underflow")
            return None
        return self.queue[0]

    def display(self):
        if self.isempty():
            print("Queue is empty")
        else:
            print("Queue:", self.queue)
```

```
    def contains(self, x):
        return x in self.queue

# Menu-driven program
q = CustomQueue()

while True:
    print("\nQueue with contains(x) operation")
    print("1. Enqueue")
    print("2. Dequeue")
    print("3. Peek")
    print("4. Display")
    print("5. Check if element exists (contains)")
    print("6. Exit")

    choice = int(input("Enter your choice: "))

    if choice == 1:
        num = int(input("Enter data: "))
        q.enqueue(num)
    elif choice == 2:
        val = q.dequeue()
        if val is not None:
            print("Dequeued:", val)
    elif choice == 3:
        val = q.peek()
        if val is not None:
            print("Front item:", val)
    elif choice == 4:
        q.display()
    elif choice == 5:
        x = int(input("Enter element to check: "))
        if q.contains(x):
            print(f"{x} exists in the queue.")
        else:
            print(f"{x} not found in the queue.")
    elif choice == 6:
        print("Exiting...")
        break
    else:
        print("Invalid choice, try again.")


Queue with contains(x) operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if element exists (contains)
6. Exit
```

```
Enter your choice:  1
Enter data:  1


Queue with contains(x) operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if element exists (contains)
6. Exit

Enter your choice:  1
Enter data:  2


Queue with contains(x) operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if element exists (contains)
6. Exit

Enter your choice:  1
Enter data:  3


Queue with contains(x) operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if element exists (contains)
6. Exit

Enter your choice:  2

Dequeued: 1

Queue with contains(x) operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if element exists (contains)
6. Exit

Enter your choice:  3

Front item: 2
```

```
Queue with contains(x) operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if element exists (contains)
6. Exit

Enter your choice:  4

Queue: [2, 3]

Queue with contains(x) operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if element exists (contains)
6. Exit

Enter your choice:  5
Enter element to check:  1

1 not found in the queue.

Queue with contains(x) operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if element exists (contains)
6. Exit

Enter your choice:  5
Enter element to check:  2

2 exists in the queue.

Queue with contains(x) operation
1. Enqueue
2. Dequeue
3. Peek
4. Display
5. Check if element exists (contains)
6. Exit

Enter your choice:  6

Exiting...
```