

Sir Padampat Singhanian University, Udaipur.



Academic Year 2023 -24

Department: School of Liberal Studies

Name of Assignment: Queue.

Full Name: Naman Jain

Roll No.: 23MCA00031

Subject: Advanced Data Structures

Date of Submission: 25/09/2023

1. Implement Queue ADT using circular Array.

```
class CircularQueue:
    def __init__(self, capacity):
        self.capacity = capacity
        self.queue = [None] * capacity
        self.front = self.rear = -1

    def is_empty(self):
        return self.front == -1

    def is_full(self):
        return (self.rear + 1) % self.capacity == self.front

    def enqueue(self, item):
        if self.is_full():
            print("Queue is full. Cannot enqueue.")
        else:
            if self.is_empty():
                self.front = self.rear = 0
            else:
                self.rear = (self.rear + 1) % self.capacity
            self.queue[self.rear] = item
            print(f"Enqueued: {item}")

    def dequeue(self):
        if self.is_empty():
            print("Queue is empty. Cannot dequeue.")
        else:
            item = self.queue[self.front]
            if self.front == self.rear:
                self.front = self.rear = -1
            else:
                self.front = (self.front + 1) % self.capacity
            return item

    def peek(self):
        if self.is_empty():
            print("Queue is empty.")
            return None
```

```

        else:
            return self.queue[self.front]

    def display(self):
        if self.is_empty():
            print("Queue is empty.")
        else:
            print("Queue elements:", end=" ")
            i = self.front
            while True:
                print(self.queue[i], end=" ")
                if i == self.rear:
                    break
                i = (i + 1) % self.capacity
            print()

queue = CircularQueue(6)
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
queue.display()

print("Peek:", queue.peek())

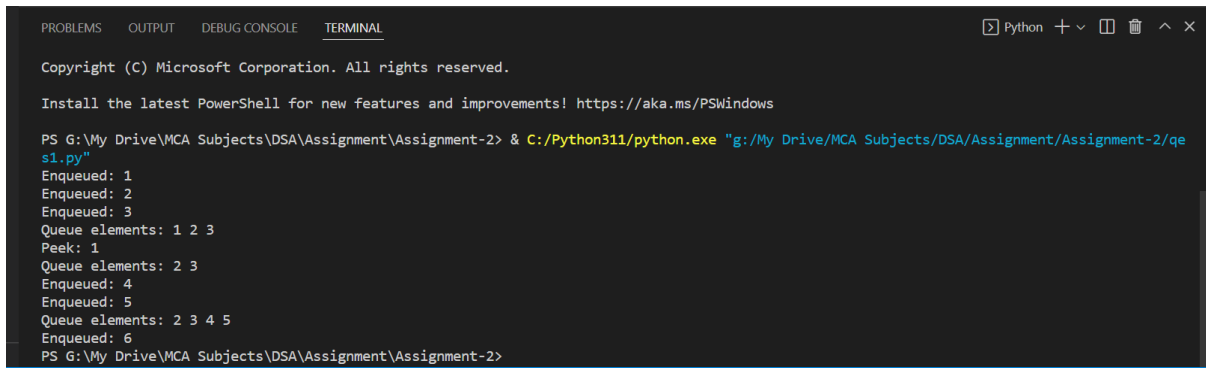
queue.dequeue()
queue.display()

queue.enqueue(4)
queue.enqueue(5)
queue.display()

queue.enqueue(6)

```

❖ Output of this Program



```
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS G:\My Drive\MCA Subjects\DSA\Assignment\Assignment-2> & C:/Python311/python.exe "g:/My Drive/MCA Subjects/DSA/Assignment/Assignment-2/qs1.py"
Enqueued: 1
Enqueued: 2
Enqueued: 3
Queue elements: 1 2 3
Peek: 1
Queue elements: 2 3
Enqueued: 4
Enqueued: 5
Queue elements: 2 3 4 5
Enqueued: 6
PS G:\My Drive\MCA Subjects\DSA\Assignment\Assignment-2>
```

2. Implement Queue ADT using a linked list.

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

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

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

    def enqueue(self, item):
        new_node = Node(item)
        if self.is_empty():
            self.front = self.rear = new_node
        else:
            self.rear.next = new_node
            self.rear = new_node

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

```

        print("Queue is empty. Cannot dequeue.")
        return None
    else:
        item = self.front.data
        self.front = self.front.next
        if self.front is None:
            self.rear = None
        return item

def peek(self):
    if self.is_empty():
        print("Queue is empty.")
        return None
    else:
        return self.front.data

def display(self):
    if self.is_empty():
        print("Queue is empty.")
    else:
        print("Queue elements:", end=" ")
        current = self.front
        while current:
            print(current.data, end=" ")
            current = current.next
        print()

if __name__ == "__main__":
    queue = Queue()
    queue.enqueue(1)
    queue.enqueue(2)
    queue.enqueue(3)
    queue.display()

    print("Peek:", queue.peek())

    queue.dequeue()
    queue.display()

    queue.enqueue(4)

```

```
queue.enqueue(5)
queue.display()
```

❖ output of program:

```
PS G:\My Drive\MCA Subjects\DSA\Assignment\Assignment-2> & C:/Python311/python.exe "g:/My Drive/MCA Subjects/DSA/Assignment/Assignment-2/qs2.py"
Queue elements: 1 2 3
Peek: 1
Queue elements: 2 3
Queue elements: 2 3 4 5
PS G:\My Drive\MCA Subjects\DSA\Assignment\Assignment-2> █
```

Perform following operations on above implemented queue and print the output of Front() method and the remaining elements in the queue:

- a. enqueue(1)
- b. enqueue(7)
- c. dequeue()
- d. enqueue(16)
- e. enqueue(5)
- f. dequeue()
- g. front()

```
from qes2 import Queue

queue = Queue()

def display_queue(queue):
    print("Front:", queue.peek())
    queue.display()
    print()

# a. enqueue(1)
queue.enqueue(1)
display_queue(queue)

# b. enqueue(7)
queue.enqueue(7)
display_queue(queue)
```

```
# c. dequeue()
queue.dequeue()
display_queue(queue)

# d. enqueue(16)
queue.enqueue(16)
display_queue(queue)

# e. enqueue(5)
queue.enqueue(5)
display_queue(queue)

# f. dequeue()
queue.dequeue()
display_queue(queue)

# g. front()
print("Front:", queue.peek())
```

❖ output of program:

```
Front: 1
Queue elements: 1 7

Front: 7
Queue elements: 7

Front: 7
Queue elements: 7 16

Front: 7
Queue elements: 7 16 5

Front: 16
Queue elements: 16 5

Front: 16
PS G:\My Drive\MCA Subjects\DSA\Assignment\Assignment-2> █
```

Write a program to implement a Stack ADT using Queue.

```
from queue import Queue

class StackUsingQueue:
    def __init__(self):
        self.queue1 = Queue()
        self.queue2 = Queue()

    def is_empty(self):
        return self.queue1.empty()

    def push(self, item):
        # Push the item to queue1
        self.queue1.put(item)

    def pop(self):
        if self.is_empty():
            print("Stack is empty. Cannot pop.")
            return None

        while self.queue1.qsize() > 1:
            self.queue2.put(self.queue1.get())

        popped_item = self.queue1.get()

        self.queue1, self.queue2 = self.queue2, self.queue1

        return popped_item

    def top(self):
        if self.is_empty():
            print("Stack is empty.")
            return None

        top_item = None
        while not self.queue1.empty():
            top_item = self.queue1.get()
```



```

        self.queue2.put(top_item)

        self.queue1, self.queue2 = self.queue2, self.queue1

    return top_item

stack = StackUsingQueue()

stack.push(10)
stack.push(20)
stack.push(30)

print("Top:", stack.top())
print("Popped:", stack.pop())

print("Top:", stack.top())
print("Popped:", stack.pop())
print("Popped:", stack.pop())

```

❖ output of program:

```

Popped: 10
PS G:\My Drive\MCA Subjects\DSA\Assignment\Assignment-2> & C:/Python311/python.exe "g:/My Drive/MCA Subjects/DSA/Assignment/Assignment-2/qes
4.py"
Top: 30
Popped: 30
Top: 20
Popped: 20
Popped: 10
PS G:\My Drive\MCA Subjects\DSA\Assignment\Assignment-2> 

```

Write a program to implement a Queue ADT using Stack.

```

class QueueUsingStack:
    def __init__(self):
        self.stack_enqueue = []
        self.stack_dequeue = []

    def is_empty(self):

```

```

        return len(self.stack_enqueue) == 0 and len(self.stack_dequeue) == 0

    def enqueue(self, item):
        self.stack_enqueue.append(item)

    def dequeue(self):
        if self.is_empty():
            print("Queue is empty. Cannot dequeue.")
            return None

        if not self.stack_dequeue:
            # Move all items from stack_enqueue to stack_dequeue
            while self.stack_enqueue:
                self.stack_dequeue.append(self.stack_enqueue.pop())

        return self.stack_dequeue.pop()

queue = QueueUsingStack()

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

print("Dequeued:", queue.dequeue())
print("Dequeued:", queue.dequeue())
print("Dequeued:", queue.dequeue())

```

❖ output of program:

```

PS G:\My Drive\MCA Subjects\DSA\Assignment\Assignment-2> & C:/Python311/python.exe "g:/My Drive/MCA Subjects/DSA/Assignment/Assignment-2/qes
7.py"
Dequeued: 1
Dequeued: 2
Dequeued: 3
PS G:\My Drive\MCA Subjects\DSA\Assignment\Assignment-2>

```

Write a program to detect a loop in a linked list.

```

class Node:
    def __init__(self, data):

```

```

        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def append(self, data):
        new_node = Node(data)
        if not self.head:
            self.head = new_node
            return
        last = self.head
        while last.next:
            last = last.next
        last.next = new_node

    def detect_loop(self):
        if not self.head:
            return False

        slow_ptr = self.head
        fast_ptr = self.head

        while fast_ptr and fast_ptr.next:
            slow_ptr = slow_ptr.next
            fast_ptr = fast_ptr.next.next

            if slow_ptr == fast_ptr:
                return True

        return False

linked_list = LinkedList()
linked_list.append(1)
linked_list.append(2)
linked_list.append(3)
linked_list.append(4)

linked_list.head.next.next.next.next = linked_list.head.next

```

```
has_loop = linked_list.detect_loop()
if has_loop:
    print("Loop detected in the linked list")
else:
    print("No loop detected in the linked list")
```

Output :

```
PS G:\My Drive\MCA Subjects\DSA\Assignment\Assignment-2> & C:/P
6.py"
Loop detected in the linked list
PS G:\My Drive\MCA Subjects\DSA\Assignment\Assignment-2> █
```

7. Which of the following is true about linked list implementation of queue?

A: In push operation, if new nodes are inserted at the beginning of the linked list, then in pop operation, nodes must be removed from end.

B: In push operation, if new nodes are inserted at the end, then in pop operation, nodes must be removed from the beginning.

C: Both of the above

D: None of the above

Ans . B