



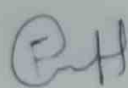
**IARE**  
INSTITUTE OF  
AERONAUTICAL ENGINEERING  
(An Autonomous Institute affiliated to JNTU, Hyderabad)  
Dundigal, Hyderabad - 500 043

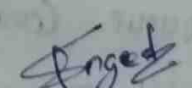
## LABORATORY WORK BOOK

Name of the Student: Muhamd Fauzan Zohair  
Class: B.Tech CSE-C Semester: II  
Course Code: ACSD06 Course Name: PPS Lab  
Name of the Course Faculty: Dr. M. Madhusudhan Reddy Faculty ID: IARE 10881  
Exercise Number: 05 Week Number: 05 Date:

Roll Number									
2	3	9	5	1	A	0	5	5	H

Exercise Number		Week Number		MARKS AWARDED					
S. No.	Exercise Number	EXERCISE NAME	Aim/ Preparation	Algorithm / Procedure		Source Code	Program Execution	Viva - Voce	Total
			Performance in the Lab	Calculations and Graphs	Results and Error Analysis	4	4	20	
									4
1	5.1	Linear Queue using list	4	4	4	4	4	4	20
2	5.2	Stack using queues							
3	5.3	Implement Queue							
4		using stacks							
5	5.4	circular Queue.							
6									
7									
8									
9									
10									
11									
12									

  
Signature of the Student

  
Signature of the Faculty

## 4.1 Linear Queue

program:-

front = 0

rear = 0

my max = 5

def create\_queue():

    queue = [] # empty

    return queue

def is\_empty(queue):

    return len(queue) == 0

def enqueue(queue, item):

    global rear, my max

    if rear < my max:

        queue.append(item)

        rear += 1

        print("Element enqueued: ", item)

else:

    print("queue is full. Element cannot be enqueued.")

def dequeue(queue):

    global front, rear

    if not is\_empty(queue):

        front += 1

        print("Element dequeued - queue, Pop(0)")

else:

    print("queue is empty cannot dequeue")

# Driver code

queue = create\_queue()

while True:

```

print("1. Enqueue")
print("2. Dequeue")
print("3. Display")
print("4. quit")
choice = int(input("Enter your choice: "))
if choice == 1:
    item = input("Enter elements to enqueue:")
    enqueue(queue, item)
elif choice == 2:
    dequeue(queue)
elif choice == 3:
    print("queue:", queue)
elif choice == 4:
    print("Exiting---")
    break.
else:
    print("Invalid choice. Please enter a valid option")

```

Exercise 4.2. stack using queue :-

Program:-

from collections import deque

class stack using queue:

def \_\_init\_\_(self):

self.queue = deque()

def push(self, x: str) -> None:

self.queue.append(x)

for i in range(len(self.queue)-1):

self.queue.append(self.queue.popleft())

```
def pop(self) → st:
```

```
    if self.queue:
```

```
        return self.queue.popleft()
```

```
    else:
```

```
        print("stack is empty, cannot pop")
```

```
        return None
```

```
def top(self) → st:
```

```
    if self.queue:
```

```
        return self.queue[-1]
```

```
    else:
```

```
        print("stack is empty. No top element")
```

```
        return None
```

```
def Empty(self) → bool:
```

```
    return len(self.queue) == 0
```

```
stack = StackUsingQueue()
```

```
while True:
```

```
    print("1. push")
```

```
    print("2. pop")
```

```
    print("4. check if empty")
```

```
    print("5. Exit")
```

```
choice = input("Enter choice")
```

```
if choice == '1':
```

```
    value = input("Enter value to push: ")
```

```
    stack.push(value)
```

```
elif choice == '2':
```

```
    p = stack.pop()
```

```
    if p is not None:
```

```
        print("popped value : p")
```

```

elif choice == '3':
    t = stack.top()
    if t is not None:
        print("Top element is ", t)
elif choice == '4':
    if stack.empty():
        print("Top stack is empty")
    else:
        print("Stack is not empty")
elif choice == '5':
    print("Exiting...")
else:
    print("Invalid choice")

```

### Q.3 Implement Queue using stack:-

Program:-

```

class QueueUsingStack:
    def __init__(self):
        self.stack1 = []
        self.stack2 = []
    def enqueue(self, x: int) -> None:
        self.stack2.append(x)
    def dequeue(self) -> int:
        if not self.stack2:
            while self.stack1:
                self.stack2.append(self.stack1.pop())
        return self.stack2.pop()

```

```

self.stack2.append(self.stack1.pop())
return self.stack2.pop()

def is-empty(self) → bool:
    return not self.stack1 and not self.stack2

queue = queue using stack()

while True:
    print("1. Enqueue")
    print("2. Dequeue")
    print("3. check if empty")
    print("4. Exit")

    choice = input()

    if choice == "1":
        value = int(input("Enter value to Enqueue: "))
        queue.enqueue(value)

    elif choice == "2":
        if queue.is-empty():
            print("Queue is empty: cannot dequeue")
        else:
            dequeue = queue.dequeue()
            print("Queue is empty: cannot dequeue")

    else:
        dq = queue.dequeue()
        print("Dequeued values: ", dq)

    elif choice == "3":

```



```

if queue_is_empty():
    print("queue is empty")
else:
    print("queue is not empty")
elif choice == '4':
    print("Exiting....")
    break
else:
    print("Invalid choice")

```

#### 4.4 Circular Queue:-

program:-

class circular queue:

def \_\_init\_\_(self, capacity):

self.capacity = capacity

self.queue = [None] \* capacity

self.front = self.rear == 1

def enqueue (self, x):

if (self.rear + 1) % self.capacity == self.front:

print("Queue is full ! cannot enqueue")

return

elif self.front == -1

self.front = 0

self.rear = (self.rear + 1) % self.capacity

self.queue[self.rear] = x

print("Enqueue element: 7/16, x).

```

def dequeue (self):
    if self.front == -1:
        print ("Queue is Empty")
        return None

    x = self.queue (self.front)

    if self.front == self.rear:
        self.front = self.rear = -1

    else:
        self.front = (self.front + 1) % self.capacity
    print ("Dequeued element : ", x)

    return x

capacity = int (input ())
queue = circular_queue (capacity)

while True:
    print ("1. Enqueue")
    print ("2. Dequeue")
    print ("3. Exit")
    e = input ()

    if e == '1':
        c = input ()
        if c == '1':
            e = input ("Enter element to enqueue : ")
            queue = enqueue (e)

    elif c == '2':
        queue.dequeue ()

    elif c == '3':

```



```
print ("Exiting --- ")
```

```
break.
```

```
else:
```

```
print ("Invalid choice").
```

4.1 Linear Queueprogram:

```

n = int(input())
command = list(map(str, input().split(' ')))
values = list(map(str, input().split(' ')))
q = []
for i in range(len(command)):
    if command[i] == 'add':
        q.append(values[i])
    elif command[i] == 'size':
        print(len(q))
    elif command[i] == 'dequeue':
        q.pop(0)
    elif command[i] == 'print':
        for j in q:
            print(j, end = ' ')

```

input

6

add, add, add, size, dequeue, print

1, 3, 4, NULL, NULL, NULL

output

accepted.

4.2 Stack using queues:

```
qsize = int(input())
```

```
s = []
```

```
command = list(map(str, input().split(' ')))
```

```
value = list(map(str, input().split(' ')))
```

```
for i in range(len(command)):
```

```
    if command[i] == 'add':
```

```
        if len(s) < qsize:
```

```
            s.append(value[i])
```

```
        else:
```

```
            print('queue is full')
```

```
    elif command[i] == 'pop':
```

```
        s.pop()
```

```
    elif command[i] == 'size':
```

```
        print(len(s))
```

```
    elif command[i] == 'print':
```

```
        for j in range(len(s)):
```

```
            print(s[len(s)-1-j], end = ' ')
```

input:

6

add, pop, add, size, print, pop

1, NULL, 3, NULL, NULL

output:

accepted.

4.3 Implement queue using stackprogram

```

n = int(input())
if n == 0:
    print('0')
for i in range(1, n+1):
    b = ""
    temp = 1
    while temp:
        if temp % 2 == 1:
            b = '1' + b
        else:
            b = '0' + b
        temp = temp // 2
    print(b)

```

inputInput :-

2

output

1

10