**#Expt18**

**#Program to implement queue using list**

q=[]

def Enqueue():

if len(q)==size: # check wether the stack is full or not

print("Queue is Full!!!!")

else:

element=input("Enter the element:")

q.append(element)

print(element,"is added to the Queue!")

def dequeue():

if not q:# or if len(stack)==0

print("Queue is Empty!!!")

else:

e=q.pop(0)

print("element removed!!:",e)

def display():

print(q)

size=int(input("Enter the size of Queue:"))

while True:

print("Select the Operation:1.Add 2.Delete 3. Display 4. Quit")

choice=int(input())

if choice==1:

Enqueue()

elif choice==2:

dequeue()

elif choice==3:

display()

elif choice==4:

break

else:

print("Invalid Option!!!")

**OUTPUT:**

Enter the size of Queue:10

Select the Operation:1.Add 2.Delete 3. Display 4. Quit

1

Enter the element:45

45 is added to the Queue!

Select the Operation:1.Add 2.Delete 3. Display 4. Quit

1

Enter the element:67

67 is added to the Queue!

Select the Operation:1.Add 2.Delete 3. Display 4. Quit

1

Enter the element:33

33 is added to the Queue!

Select the Operation:1.Add 2.Delete 3. Display 4. Quit

1

Enter the element:89

89 is added to the Queue!

Select the Operation:1.Add 2.Delete 3. Display 4. Quit

3

['45', '67', '33', '89']

Select the Operation:1.Add 2.Delete 3. Display 4. Quit

2

element removed!!: 45

Select the Operation:1.Add 2.Delete 3. Display 4. Quit

3

['67', '33', '89']

Select the Operation:1.Add 2.Delete 3. Display 4. Quit

4

**#Expt19**

**#Program to implement priority queue**

**# class for Node with data and priority**

class Node:

def \_\_init\_\_(self, info, priority):

self.info = info

self.priority = priority

# class for Priority queue

class PriorityQueue:

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

self.queue = list()

# if you want you can set a maximum size for the queue

def insert(self, node):

# if queue is empty

if self.size() == 0:

# add the new node

self.queue.append(node)

else:

# traverse the queue to find the right place for new node

for x in range(0, self.size()):

# if the priority of new node is greater

if node.priority >= self.queue[x].priority:

# if we have traversed the complete queue

if x == (self.size()-1):

# add new node at the end

self.queue.insert(x+1, node)

else:

continue

else:

self.queue.insert(x, node)

return True

def delete(self):

# remove the first node from the queue

return self.queue.pop(0)

def show(self):

for x in self.queue:

print( str(x.info)+" - "+str(x.priority))

def size(self):

return len(self.queue)

pQueue = PriorityQueue()

node1 = Node("C", 3)

node2 = Node("B", 2)

node3 = Node("A", 1)

node4 = Node("Z", 26)

node5 = Node("Y", 25)

node6 = Node("L", 12)

pQueue.insert(node1)

pQueue.insert(node2)

pQueue.insert(node3)

pQueue.insert(node4)

pQueue.insert(node5)

pQueue.insert(node6)

pQueue.show()

print("--------")

pQueue.delete()

pQueue.show()

**OUTPUT:**

**A - 1**

**B - 2**

**C - 3**

**L - 12**

**Y - 25**

**Z - 26**

**--------**

**B - 2**

**C - 3**

**L - 12**

**Y - 25**

**Z - 26**