24ucs088

Exercise 4.a

class queue:

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

self.queue=[]

def is\_empty(self):

return len(self.queue)==0

def enqueue(self,item):

self.queue.append(item)

def dequeue(self):

if self.is\_empty():

raise IndexError("queue is empty")

return self.queue.pop(0)

def peek(self):

if self.is\_empty():

raise IndexError("queue is empty")

return self.queue[0]

def size(self):

return len(self.queue)

q=queue()

q.enqueue(10)

q.enqueue(20)

q.enqueue(30)

print("queue after enqueuing elements:",q.queue)

print("dequeued elements:",q.dequeue())

print("queue after dequeuing an elements:",q.queue)

print("front elements:",q.peek())

print("queue size:",q.size())

OUTPUT

queue after enqueuing elements: [10, 20, 30]

dequeued elements: 10

queue after dequeuing an elements: [20, 30]

front elements: 20

queue size: 2

EXERCISE 4.B

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,data):

new\_node=node(data)

if self.rear is None:

self.front=self.rear=new\_node

return

self.rear.next=new\_node

self.rear=new\_node

def dequeue(self):

if self.is\_empty():

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.is\_empty():

return None

return self.front.data

def size(self):

count=0

current=self.front

while current:

count+=1

current=current.next

return count

q=queue()

q.enqueue(10)

q.enqueue(20)

q.enqueue(30)

print("queue elements:",end=" ")

while not q.is\_empty():

print(q.dequeue(),end=" ")

print("\n is the queue?",q.is\_empty())

OUTPUT

queue elements: 10 20 30

is the queue? True