1.

MAX\_SIZE = 10

queue = [0] \* MAX\_SIZE

head = None

rear = 0

item1 = 'Grape'

item2 = 'Mango'

item3 = 'Apple'

def enqueue1(item1):

global rear

global MAX\_SIZE

global queue

if rear == MAX\_SIZE-1:

print('佇列已滿')

else:

queue[rear] = item1

rear += 1

def enqueue2(item2):

global rear

global MAX\_SIZE

global queue

if rear == MAX\_SIZE-1:

print('佇列已滿')

else:

queue[rear] = item2

rear += 1

def enqueue3(item3):

global rear

global MAX\_SIZE

global queue

if rear == MAX\_SIZE-1:

print('佇列已滿')

else:

queue[rear] = item3

rear += 1

enqueue1(item1)

enqueue2(item2)

enqueue3(item3)

print(queue)

print(MAX\_SIZE)

2.

class food:

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

self.name = ' ' \* 20

self.next = None

head = None

rear = None

length = 1

def enqueue (name):

global head

global rear

global length

new\_data = food()

new\_data.name = name

if rear == None:

head = new\_data

else:

rear.next = new\_data

length += 1

rear = new\_data

new\_data.next = None

def dequeue ():

global head

global rear

if head == None:

print('佇列為空!')

else:

print(head.name)

head = head.next

enqueue('漢堡')

enqueue('薯條')

enqueue('可樂')

enqueue('炸雞')

print(length)

dequeue()

dequeue()

dequeue()

dequeue()

3.

class Node:

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

self.data = data

self.next = None

self.prev = None

class Queue:

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

self.head = None

self.last = None

def enqueue(self, data):

if self.last is None:

self.head = Node(data)

self.last = self.head

else:

self.last.next = Node(data)

self.last.next.prev=self.last

self.last = self.last.next

def dequeue(self):

if self.head is None:

return None

else:

temp = self.head.data

self.head = self.head.next

self.head.prev = None

return temp

def printqueue(self):

temp=self.last

while temp is not None:

print(temp.data)

temp=temp.prev

queue = Queue()

queue.enqueue(1)

queue.enqueue(2)

queue.enqueue(3)

queue.enqueue(4)

queue.enqueue(5)

queue.enqueue(6)

queue.enqueue(7)

queue.printqueue()

4.

MAX\_SIZE = 20

stack = [0] \* MAX\_SIZE

top = 0

def push(data):

global top

global MAX\_SIZE

global stack

if top >= MAX\_SIZE-1:

print('堆疊已滿，無法再加入')

return None

else:

top += 1

stack[top] = data

def pop():

global top

global stack

if isEmpty():

print('堆疊為空')

return None

else:

print(stack[top])

top -= 1

return stack[top+1]

def isEmpty():

if top == -1:

return True

else:

return False

push(1)

push(2)

push(3)

push(4)

push(5)

push(6)

push(7)

pop()

pop()

pop()

pop()

pop()

pop()

pop()

5.

class Node:

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

self.data = 0

self.next = None

top = None

length = 0

def isEmpty():

global top

if(top == None):

return True

else:

return False

def push(data):

global top

global length

new\_add\_node = Node()

new\_add\_node.data = data

new\_add\_node.next = top

top = new\_add\_node

length += 1

def pop():

global top

if isEmpty():

print('目前為空堆疊')

return -1

else:

ptr = top

top = top.next

temp = ptr.data

print(temp)

return temp

push('漢堡')

push('薯條')

push('可樂')

push('炸雞')

print(length)

pop()

pop()

pop()

pop()