第3次隨堂-隨堂-QZ3

> 學號：112111220
> 姓名：葉崇聖
> 作業撰寫時間：180 (mins，包含程式撰寫時間)
> 最後撰寫文件日期：2023/11/24

本份文件包含以下主題：(至少需下面兩項，若是有多者可以自行新增)

- ☑ 說明內容
- ☑ 個人認為完成作業須具備觀念

說明程式與內容

開始寫說明，該說明需說明想法， 並於之後再對上述想法的每一部分將程式進一步進行展現， 若需引用程式區則使用下面方法， 若為.cs檔內程式除了於敘述中需註明檔案名稱外， 還需使用語法```語言種類 程式碼 ```，其中語言種類若是要用python則使用py，java則使用java，C/C++則使用cpp， 下段程式碼為語言種類選擇csharp使用後結果：

```
public void mt_getResult(){
    ...
}
```

若要於內文中標示部分網頁檔，則使用以下標籤```html 程式碼 ```， 下段程式碼則為使用後結果：

```
<%@ Page Language="C#" AutoEventWireup="true" ...>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" ...>
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
        </div>
    </form>
</body>
</html>
```

更多markdown方法可參閱https://ithelp.ithome.com.tw/articles/10203758

請在撰寫"說明程式與內容"該塊內容，請把原該塊內上述敘述刪除，該塊上述內容只是用來指引該怎麼撰寫內容。

1. 請參閱投影片Topic5的第31至35頁，請用物件導向方式進行新增與刪除。 (請參照題目pdf)

Ans:

```
class Node:  # 節點類別
    def __init__(self, data=None):
        self.data = data  # 儲存節點的資料
        self.next = None  # 指向下一個節點的鏈結


class Stack:  # 堆疊類別
    def __init__(self):
        self.top = None  # 堆疊頂端指標

    def push(self, data):
        new_node = Node(data)  # 建立新節點
        new_node.next = self.top  # 新節點的鏈結指向目前的頂端節點
        self.top = new_node
        print(f"已將 {data} 推入堆疊。")

    def pop(self):
        if self.is_empty():
            raise Exception("堆疊為空。")  # 堆疊為空時拋出異常
        data = self.top.data  # 取得頂端節點的資料
        self.top = self.top.next
        print(f"已從堆疊彈出 {data}。")
        return data

    def peek(self):
        if self.is_empty():
            raise Exception("堆疊為空。")  # 堆疊為空時拋出異常
        return self.top.data  # 返回頂端節點的資料

    def is_empty(self):
        return self.top is None  # 當頂端為 None 時，堆疊為空


class Queue:  # 佇列類別
    def __init__(self):
        self.front = None  # 佇列前端指標
        self.rear = None  # 佇列尾端指標

    def enqueue(self, data):
        new_node = Node(data)  # 建立新節點
        if self.rear is None:  # 如果佇列為空
            self.front = new_node  # 新節點同時成為前端與尾端
            self.rear = new_node
        else:
            self.rear.next = new_node  # 尾端的鏈結指向新節點
            self.rear = new_node  # 更新尾端為新節點
        print(f"已將 {data} 加入佇列。")
```

```
    def dequeue(self):
        if self.is_empty():
            raise Exception("佇列為空。")  # 佇列為空時拋出異常
        data = self.front.data  # 取得前端節點的資料
        self.front = self.front.next  # 移除前端節點，更新前端為下一節點
        if self.front is None:  # 如果前端變為空
            self.rear = None  # 同時更新尾端為 None
        print(f"已從佇列取出 {data}。")
        return data

    def peek(self):
        if self.is_empty():
            raise Exception("佇列為空。")  # 佇列為空時拋出異常
        return self.front.data  # 返回前端節點的資料

    def is_empty(self):
        return self.front is None  # 當前端為 None 時，佇列為空


# 測試堆疊
print("堆疊測試：")
stack = Stack()
stack.push(10)
stack.push(20)
stack.push(30)
print("堆疊頂端元素:", stack.peek())
stack.pop()
print("堆疊是否為空?", stack.is_empty())
print("")

# 測試佇列
print("佇列測試：")
queue = Queue()
queue.enqueue(1)
queue.enqueue(2)
queue.enqueue(3)
print("佇列前端元素:", queue.peek())
queue.dequeue()
print("佇列是否為空?", queue.is_empty())
```

個人認為完成作業須具備觀念

開始寫說明，需要說明本次練習需學會那些觀念 (需寫成文章，需最少50字，並且文內不得有你、我、他三種文字)且必須提供完整與練習相關過程的notion筆記連結