第3次隨堂-隨堂-QZ3

> 學號：112112105
> 姓名：李佩琪
> 作業撰寫時間：55 (mins，包含程式撰寫時間)
> 最後撰寫文件日期：2024/12/

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，用於儲存資料及指向下一個節點的鏈結。
    """
    def __init__(self, data=None):
        self.data = data  # 節點儲存的資料
        self.next = None  # 指向下一個節點的鏈結

class Stack:
    """
    堆疊類別 (Stack)，實現後進先出 (LIFO) 的資料結構。
    """
    def __init__(self):
        self.top = None  # 堆疊頂端的節點

    def push(self, data: int):
        """
        將資料加入堆疊頂端。
        """
        new_node = Node(data)  # 建立新節點
        new_node.next = self.top  # 新節點的 next 指向當前的堆疊頂端
        self.top = new_node  # 更新堆疊頂端為新節點

    def pop(self) -> int:
        """
        從堆疊移除頂端節點並返回其資料。
        若堆疊為空，則拋出例外。
        """
        if self.top is None:
            raise Exception("Stack is empty.")  # 堆疊空時拋出異常
        removed_node = self.top  # 取出堆疊頂端的節點
        self.top = self.top.next  # 更新頂端為下一個節點
        data = removed_node.data  # 儲存移除節點的資料
        del removed_node  # 釋放記憶體
        return data  # 返回移除節點的資料

class Queue:
    """
    佇列類別 (Queue)，實現先進先出 (FIFO) 的資料結構。
    """
    def __init__(self):
        self.front = None  # 佇列的前端
        self.rear = None  # 佇列的後端

    def enqueue(self, data: int):
        """
        將資料加入佇列後端。
        """
        new_node = Node(data)  # 建立新節點
```

```
        if self.rear is None:
            # 若佇列為空，前後端都指向新節點
            self.front = new_node
            self.rear = new_node
        else:
            # 將後端節點的 next 指向新節點，並更新後端為新節點
            self.rear.next = new_node
            self.rear = new_node

    def dequeue(self) -> int:
        """
        從佇列移除前端節點並返回其資料。
        若佇列為空，則拋出例外。
        """
        if self.front is None:
            raise Exception("Queue is empty.")  # 佇列空時拋出異常
        removed_node = self.front  # 取出佇列前端的節點
        self.front = self.front.next  # 更新前端為下一個節點
        if self.front is None:
            # 若佇列已空，後端也設為 None
            self.rear = None
        data = removed_node.data  # 儲存移除節點的資料
        del removed_node  # 釋放記憶體
        return data  # 返回移除節點的資料
```

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