第3次作業-作業-HW3

> 學號：1234567
> 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：2023/11/27

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. 請回答下面問題。

Ans: P.19

```
def top_item(stack, top):
    if top == -1:  # 如果top指標為-1，代表堆疊為空
        return "Stack is empty"  # 堆疊為空
    else:
        return stack[top]  # 否則，返回堆疊頂端的元素
```

P.21

```
def is_full(stack, top, N):
    if top == N - 1:  # 如果top指標指向堆疊的頂端，即堆疊已滿
        return True  # 傳回True，表示堆疊已滿
    else:
        return False  # 否則傳回False，表示堆疊未滿
```

2. 請回答下面問題。

Ans:

```
def knight_tour(N, startX, startY):
    # 騎士的八個可能移動方向
    moves = [
        (-2, 1), (-2, -1), (2, 1), (2, -1),
        (1, 2), (1, -2), (-1, 2), (-1, -2)
    ]

    # 初始化棋盤，False 表示尚未訪問
    board = [[False for _ in range(N)] for _ in range(N)]

    # 堆疊，儲存目前的位置和步驟
    stack = [(startX, startY)]
    board[startX][startY] = True  # 起點已訪問
    visited_count = 1  # 已訪問格數

    while stack:
        x, y = stack[-1]  # 取堆疊頂部的當前位置
        found_next_move = False

        for dx, dy in moves:
            nx, ny = x + dx, y + dy
            # 如果新位置在棋盤範圍內且未被訪問
            if 0 <= nx < N and 0 <= ny < N and not board[nx][ny]:
                stack.append((nx, ny))  # 將新位置加入堆疊
                board[nx][ny] = True  # 標記為已訪問
                visited_count += 1
```

```
                found_next_move = True
                break  # 跳出迴圈，繼續探索新位置

        if not found_next_move:
            # 如果無法繼續移動，則回溯
            stack.pop()

    # 判斷是否訪問了棋盤上的所有格子
    return visited_count == N * N


# 主程式：接受輸入並輸出結果
if __name__ == "__main__":
    # 輸入三個數字：N, startX, startY
    N = int(input("請輸入棋盤大小 N (4 ≤ N ≤ 10): "))
    startX = int(input("請輸入騎士的起始位置 X (0 ≤ X < N): "))
    startY = int(input("請輸入騎士的起始位置 Y (0 ≤ Y < N): "))

    # 確保輸入範圍合法
    if 4 <= N <= 10 and 0 <= startX < N and 0 <= startY < N:
        # 執行騎士遍歷棋盤的函式
        result = knight_tour(N, startX, startY)
        print(result)  # 輸出 True 或 False
    else:
        print("輸入數據不合法，請重新確認。")
```

3. 請回答下面問題：

Ans:

```
def josephus_problem(n, k):
    # 建立一個初始隊列，從 1 到 n
    people = list(range(1, n + 1))

    # 初始化索引指向第一個人
    index = 0

    # 不斷移除直到只剩下一個人
    while len(people) > 1:
        # 計算要移除的人的索引
        index = (index + k - 1) % len(people)
        # 移除該人
        people.pop(index)

    # 剩下的最後一個人即為勝利者
    return people[0]

# 主程式：接受輸入並輸出結果
if __name__ == "__main__":
    n = int(input("請輸入參與遊戲的人數 n: "))
    k = int(input("請輸入每次數到第 k 個人的值 k: "))
```

```
    # 確保輸入值是正整數
    if n > 0 and k > 0:
        winner = josephus_problem(n, k)
        print(f"最後剩下的人的編號是: {winner}")
    else:
        print("輸入的 n 和 k 必須是正整數，請重新確認。")
```

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

1.學會如何活用python實作了堆疊的create、push和pop，來進行操作。 2.利用迴圈語法進行""約瑟夫斯問題""的數學理解。