# 第3次作業-作業-HW3

> 學號：112112105
> 姓名：李佩琪
> 作業撰寫時間：65 (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. 請回答下面問題。

Ans:

```
def is_full(stack, top, capacity):
    """
    判斷堆疊是否已滿。
    :param stack: 堆疊
    :param top: 堆疊頂端的索引
    :param capacity: 堆疊的容量
    :return: 如果已滿返回 True，否則返回 False
    """
    return top == capacity - 1  # 如果頂端索引等於容量減一，則堆疊已滿

def is_empty(stack, top):
    """
    判斷堆疊是否為空。
    :param stack: 堆疊
    :param top: 堆疊頂端的索引
    :return: 如果為空返回 True，否則返回 False
    """
    return top == -1  # 如果頂端索引為 -1，則堆疊為空
```

2. 請回答下面問題。

Ans:

```
def is_valid_move(x, y, N, visited):
    """
    檢查是否在棋盤內且該格子未被訪問。
    :param x: 當前位置的行座標
    :param y: 當前位置的列座標
    :param N: 棋盤大小
    :param visited: 記錄已訪問的棋盤格
    :return: 如果可以訪問返回 True，否則返回 False
    """
    return 0 <= x < N and 0 <= y < N and not visited[x][y]

def knight_tour(N, startX, startY):
    """
    使用深度優先搜尋 (DFS) 解決騎士巡邏問題。
    :param N: 棋盤大小
    :param startX: 起始位置的行座標
    :param startY: 起始位置的列座標
    :return: 如果存在解返回 True，否則返回 False
    """
    # 騎士的 8 個可能移動方向
    moves = [(-2, -1), (-2, 1), (-1, -2), (-1, 2),
             (1, -2), (1, 2), (2, -1), (2, 1)]
    visited = [[False] * N for _ in range(N)]  # 初始化棋盤訪問記錄
```

```
    def dfs(x, y, visited_count):
        """
        深度優先搜尋，用於嘗試每個可能的路徑。
        :param x: 當前位置的行座標
        :param y: 當前位置的列座標
        :param visited_count: 已訪問格子的數量
        :return: 如果找到解返回 True，否則返回 False
        """
        # 訪問當前格子
        visited[x][y] = True
        visited_count += 1

        # 如果所有格子都被訪問過，則成功
        if visited_count == N * N:
            return True

        # 嘗試所有可能的移動方向
        for dx, dy in moves:
            nx, ny = x + dx, y + dy
            if is_valid_move(nx, ny, N, visited):
                if dfs(nx, ny, visited_count):
                    return True

        # 如果無法繼續，回溯
        visited[x][y] = False
        return False

    # 從起始位置開始搜尋
    return dfs(startX, startY, 0)

# 主程式入口
if __name__ == "__main__":
    N = int(input("輸入棋盤大小 N: "))  # 棋盤大小
    startX, startY = map(int, input("輸入起始位置 (startX startY): ").split())  # 起始位置

    if knight_tour(N, startX, startY):
        print("找到解：True")
    else:
        print("無解：False")
```

3. 請回答下面問題：

Ans:

```
def josephus(n, k):
    """
    解決約瑟夫問題，找出最後存活的人的編號。
    :param n: 總人數
    :param k: 每次刪除第 k 個人
    :return: 最後存活的人的編號
    """
```

```
    # 初始化所有人的編號，從 1 到 n
    people = list(range(1, n + 1))
    index = 0  # 記錄當前需要刪除的位置

    # 模擬過程，直到只剩下一個人
    while len(people) > 1:
        # 計算下一個被刪除的位置，考慮環狀列表
        index = (index + k - 1) % len(people)
        people.pop(index)  # 刪除該位置的人

    # 返回最後存活的人的編號
    return people[0]

# 主程式入口
if __name__ == "__main__":
    n, k = map(int, input("輸入總人數 n 和間隔數 k（以空格分隔）：").split())
    print(f"最後存活的人的編號是：{josephus(n, k)}")
```

## 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

完成本次作業需要掌握多個核心概念。首先，需要理解迴圈與條件判斷在程式執行過程中的重要性，尤其是解決約瑟夫問題時，如何運用模運算進行環狀數據處理