第3次作業-作業-HW3

學號：112111207
姓名：陳品霖
作業撰寫時間：60 (mins，包含程式撰寫時間)
最後撰寫文件日期：2024/11/25

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

1.

Ans:

```
def IsFull(Stack,top,N):
    if top == N-1:
        return True;
    else:
        return False;
def Isempty(Stack,top,N):
    if top == -1:
        return True;
    else:
        return False;
```

2.

Ans:

```
def knight_tour(N, startX, startY):
    # 定義騎士的8個移動方向
    dx = [-2, -1, 1, 2, 2, 1, -1, -2]
    dy = [-1, -2, -2, -1, 1, 2, 2, 1]

    # 初始化棋盤，所有格子標記為未訪問
    board = [[False for _ in range(N)] for _ in range(N)]

    # 初始化堆疊
    stack = [(startX, startY)]
    board[startX][startY] = True  # 標記起始位置已訪問
    visited_count = 1  # 計算訪問過的格子數

    while stack:
        x, y = stack.pop()
```

```
        # 嘗試8個方向
        for i in range(8):
            nx, ny = x + dx[i], y + dy[i]

            # 確保新位置在棋盤內，且未被訪問
            if 0 <= nx < N and 0 <= ny < N and not board[nx][ny]:
                board[nx][ny] = True  # 標記為已訪問
                stack.append((nx, ny))
                visited_count += 1
                break  # 進入下一個位置後，重新開始搜尋

    # 若訪問格子的數量等於棋盤大小，表示成功遍歷
    return visited_count == N * N

# 主程式
if __name__ == "__main__":
    # 輸入
    N = int(input("輸入棋盤大小 N (4 <= N <= 10): "))
    if 4 <= N <= 10:
        startX = int(input("輸入起始位置 startX (0 <= startX < N): "))
        startY = int(input("輸入起始位置 startY (0 <= startY < N): "))

        # 檢查起始位置是否合法
        if 0 <= startX < N and 0 <= startY < N:
            result = knight_tour(N, startX, startY)
            print("True" if result else "False")
        else:
            print("起始位置無效，請確保 0 <= startX, startY < N")
    else:
        print("N 必須在範圍 4 <= N <= 10")
```

3.

Ans:

```
def josephus_problem(n, k):
    # 建立初始的圓圈 (1 到 n)
    circle = list(range(1, n + 1))
    index = 0  # 從第 1 個人開始 (0-indexed)

    # 持續移除，直到只剩下一個人
    while len(circle) > 1:
        # 計算需要移除的人的索引
        index = (index + k - 1) % len(circle)
        circle.pop(index)  # 移除該人

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

# 主程式
if __name__ == "__main__":
```

```
    # 輸入 n 和 k
    n = int(input("輸入參加遊戲的人數 n: "))
    k = int(input("輸入計數的步數 k: "))

    # 檢查輸入是否有效
    if n > 0 and k > 0:
        # 計算並輸出結果
        result = josephus_problem(n, k)
        print(f"最後存活的人的編號是: {result}")
    else:
        print("n 和 k 必須是正整數！")
```

個人認為完成作業須具備觀念

- 應熟悉資料結構中的堆疊與其操作，理解如何利用堆疊模擬特定問題的流程。

- 需掌握循環結構的應用，特別是在模擬約瑟夫斯問題時，如何利用模數運算處理環形結構。

- 應了解條件判斷與索引操作的關係，尤其是在列表操作中，確保邏輯的正確性。最後，需能將問題分解成步驟，並結合數學思維與程式解決實際問題。