# 第3次作業-作業-HW3

學號：112111219
姓名：陳恩偉
作業撰寫時間：110 (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

## 說明程式與內容

1. 請回答下面問題。

Ans:

```
class Stack:
    def __init__(self, size):
        self.stack = [None] * size
        self.top = -1
        self.max_size = size

    def push(self, item):
        if self.isFull():
            print("Stack滿,can't push")
            return
        self.top += 1
        self.stack[self.top] = item
        print(f"Pushed: {item}")

    def pop(self):
        if self.isEmpty():
            print("Stack空,can't pop")
            return None
        popped_item = self.stack[self.top]
        self.top -= 1
        print(f"Popped: {popped_item}")
        return popped_item

    def isFull(self):
        return self.top == self.max_size - 1

    def isEmpty(self):
        return self.top == -1

#test
stack = Stack(3)
stack.push('A')
```

```
stack.push('B')
stack.push('C')
stack.push('D')
stack.pop()
stack.pop()
stack.pop()
stack.pop()
```

2. 請回答下面問題。

Ans:

```
class KnightTour:
    def __init__(self, N, startX, startY):
        self.N = N
        self.board = [[-1 for _ in range(N)] for _ in range(N)]
        self.moves = [(-2, -1), (-2, 1), (-1, 2), (1, 2), (2, 1), (2, -1), (1,
-2), (-1, -2)]
        self.startX, self.startY = startX, startY

    def is_valid_move(self, x, y):
        return 0 <= x < self.N and 0 <= y < self.N and self.board[x][y] == -1

    def knight_tour(self):
        stack = [(self.startX, self.startY, 0)]
        self.board[self.startX][self.startY] = 0

        while stack:
            x, y, move_count = stack[-1]

            if move_count == self.N * self.N - 1:
                return True


            for dx, dy in self.moves:
                next_x, next_y = x + dx, y + dy
                if self.is_valid_move(next_x, next_y):
                    self.board[next_x][next_y] = move_count + 1
                    stack.append((next_x, next_y, move_count + 1))
                    break
            else:
                self.board[x][y] = -1
                stack.pop()

        return False

#test
N = int(input("輸入棋盤大小 N: "))
startX, startY = map(int, input("輸入起始位置 (startX, startY): ").split())

knight_tour = KnightTour(N, startX, startY)
```

```
result = knight_tour.knight_tour()
print(result)
```

3. 請回答下面問題：

Ans:

```
def josephus(n, k):
    if n == 1:
        return 1
    else:
        return (josephus(n - 1, k) + k - 1) % n + 1

n = 5
k = 2
result = josephus(n, k)
print(f"結果：{result}")
```

## 個人認為完成作業須具備觀念

第一題涉及堆疊的基本操作，需要掌握如何檢查堆疊是否滿或為空，這是處理資料存取順序的重要基礎。第二題是關於棋盤上的騎士巡邏問題，著重在如何使用堆疊追蹤移動路徑並進行回溯，這對於解決搜尋問題十分有幫助。第三題是經典的約瑟夫問題，要求運用遞迴概念解決圓圈淘汰問題，這題目展示了模擬複雜計數問題的技巧。這三題共同鍛鍊了對資料結構的理解、控制流程的掌握及遞迴思考方式。