第3次作業-作業-HW3

> 學號：112111234
> 姓名：阮陳家興
> 作業撰寫時間：180 (mins，包含程式撰寫時間)
> 最後撰寫文件日期：2023/09/22

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:

```
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 is full,can't push")
            return
        self.top += 1
        self.stack[self.top] = item
        print(f"Pushed: {item}")

    def pop(self):
        if self.isEmpty():
            print("Stack is empty,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('C')
stack.pop()
stack.push('D')
stack.pop()
stack.pop()
stack.pop()
stack.pop()
```

2. 請回答下面問題。

```python
def knight_tour(N, startX, startY):
    # 騎士的所有移動方向
    moves = [
        (-2, -1), (-2, +1), (-1, +2), (+1, +2),
        (+2, +1), (+2, -1), (+1, -2), (-1, -2)
    ]

    # 棋盤和stack
    visited = [[False] * N for _ in range(N)]  # 標記已造訪的儲存格
    stack = [(startX, startY)]  # 儲存步驟
    visited[startX][startY] = True
    steps = 1  # 通過的儲存格數量

    def count_valid_moves(x, y):
        """計算從位置 (x, y) 開始的合法移動次數。"""
        count = 0
        for dx, dy in moves:
            nx, ny = x + dx, y + dy
            if 0 <= nx < N and 0 <= ny < N and not visited[nx][ny]:
                count += 1
        return count

    while stack:
        x, y = stack[-1]  # 從堆疊頂部取得目前位置

        # 選擇一個有效的動作
        next_move = None
        min_degree = float('inf')  # 找出有效移動次數最少的方格

        for dx, dy in moves:
            nx, ny = x + dx, y + dy
            if 0 <= nx < N and 0 <= ny < N and not visited[nx][ny]:
                degree = count_valid_moves(nx, ny)
                if degree < min_degree:
                    min_degree = degree
                    next_move = (nx, ny)

        # 如果你找到了行動，就行動吧
        if next_move:
            nx, ny = next_move
            stack.append((nx, ny))
            visited[nx][ny] = True
            steps += 1
        else:
            # 如果沒有更多有效的動作，則回溯
            stack.pop()

    # 檢查是否所有儲存格都已通過
    return steps == N * N

# Test
N, startX, startY = map(int, input(" N, startX, startY = ").split())
print(f"結果：{knight_tour(N, startX, startY)}")
```

Ans:

3. 請回答下面問題：

Ans:

```
def eliminate(n, k):
    players = list(range(1, n + 1))  # 列出從 1 到 n
    idx = 0  # 計數起始位置

    while len(players) > 1:
        idx = (idx + k - 1) % len(players)  # 確定誰被淘汰
        players.pop(idx)  # 從清單中刪除該人

    return players[0]  #    最後留下一個人

# Test
n, k = map(int, input(" n , k = ").split())
print(f"結果：{eliminate(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