第3次作業-作業-HW3

> 學號：112111225
> 姓名：林芷羽
> 作業撰寫時間：180 (mins，包含程式撰寫時間)
> 最後撰寫文件日期：2024/11/27

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:

```
#Create
from typing import List, Tuple
def create(stack: List[str], size: int = 10) -> int:
"""
創立堆疊
Args:
    stack List [str]堆疊名稱
    size int 堆疊大小,若沒設定則預設大小為10
Returns
    (int)索引值
"""
#初始化堆疊，直接修改原始列表內容(使用切片操作清空列表，然後賦值新結構)
    stack [:] = [""] * size #創建一個大小為size的堆疊、初始為空字符
    top: int = -1 #初始時堆疊為空,Top指標設為-1
    return top
pass

#Push
from typing import List, Tuple
def push(item: str, stack: List[str], top: int) -> int:
"""
推入元素至堆疊結構
Args:
    item str欲堆入元素
    stack (List[str])堆疊名稱
    top (int)目前堆疊中的索引
Returns
    (int)索引值
"""
    size: int = len(stack)
    if top == size - 1:
        print("Stack is full.")#堆疊已滿
    else:
        top += 1 #將top指標加1
        stack[top] = item #將資料放入堆疊
    return top
pass

#Pop
from typing import List, Tuple, Union
def pop(stack: List[str], top: int) -> Tuple[Union[str, None], int]:
"""
推入元素至堆疊結構
Args:
    stack (List[str])堆疊名稱
    top (int)目前堆疊中的索引
    size (int) 堆疊大小,若沒設定則預設大小為10
Returns
    (Union[str, None])推出元素，若原本堆疊為空則推出None
```

```
    (int)索引值
"""
    if top == -1:
        print("Stack is empty.") #堆疊為空
        return None, top
    else:
        item = stack[top] #取出堆疊頂堆的資料
        stack[top] = "" #清除堆疊內內容
        top -= 1 #將top指標減1
        return item, top #返回取出的項目，堆疊和更新後的top
pass

#IsFull
def is_full(stack, top, max_size):
    if top == max_size - 1:
        return True
    else:
        return False
#Isempty
def is_empty(stack, top):
    if top == -1:
        return True
    else:
        return False
```

2. 請回答下面問題。

Ans:

```
def knight_tour(N, startX, startY):
    # 定義騎士的 8 種可能移動方式
    moves = [
        (-2, -1), (-2, 1), (2, -1), (2, 1),
        (-1, -2), (-1, 2), (1, -2), (1, 2)
    ]

    # 初始化棋盤與起始點
    board = [[False for _ in range(N)] for _ in range(N)]
    board[startX][startY] = True  # 標記起始點已訪問

    # 從起始點開始進行深度優先搜尋 (DFS)
    def dfs(x, y, step):
        # 若所有格子都已訪問，則返回 True
        if step == 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

            # 確認新位置在棋盤內，且未被訪問過
            if 0 <= nx < N and 0 <= ny < N and not board[nx][ny]:
```

```
                board[nx][ny] = True  # 標記為已訪問
                if dfs(nx, ny, step + 1):  # 深入搜尋
                    return True
                board[nx][ny] = False  # 回溯，標記為未訪問

        return False

    # 呼叫 DFS 從起始點開始
    return dfs(startX, startY, 1)


if __name__ == "__main__":
    # 使用者輸入棋盤大小與起始位置
    while True:
        try:
            N = int(input("請輸入棋盤大小 N（4 ≤ N ≤ 10）："))
            if 4 <= N <= 10:
                break
            else:
                print("無效輸入！請輸入 4 至 10 的整數。")
        except ValueError:
            print("請輸入整數！")

    while True:
        try:
            startX = int(input(f"請輸入起始位置 X 座標（0 至 {N - 1}）："))
            startY = int(input(f"請輸入起始位置 Y 座標（0 至 {N - 1}）："))
            if 0 <= startX < N and 0 <= startY < N:
                break
            else:
                print(f"無效輸入！座標必須在範圍 0 至 {N - 1} 內。")
        except ValueError:
            print("請輸入整數！")

    # 輸出結果
    if knight_tour(N, startX, startY):
        print("True")
    else:
        print("False")
```

3. 請回答下面問題：

Ans:

```
def josephus_problem(n, k):
    people = list(range(1, n + 1))  # 建立編號 1 到 n 的人
    index = 0  # 起始計數的位置

    while len(people) > 1:
        index = (index + k - 1) % len(people)  # 計算要移除的人的索引
        people.pop(index)  # 移除該人
```

```
    return people[0]  # 剩下的最後一個人

n = int(input("請輸入總人數 n: "))
k = int(input("請輸入每次報數的間隔 k: "))

result = josephus_problem(n, k)
print(f"最後留下的人是編號: {result}")
```

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

> 需要學會堆疊的基本操作，如創建堆疊、推入元素、彈出元素以及檢查堆疊是否為空或已滿，運用演算法來解決遞迴和回溯問題