國立屏東大學 資訊工程系 程式設計(一)

Turnin作業5

- chapter 13
- Turnin Code: c.hw5
- Due Date: 2026/04/29 23:59 **Hard Deadline**

繳交方式說明

本次作業繳交將以資料夾的形式繳交，需要為每一題建立一個資料夾（資料夾名稱為該題題目前方之代號，第一題為□p1□□第二題為□p2□□餘以此類推），

繳交說明可參考作業3：**連結**

任何未依照正確繳交格式的檔案將以 0 分計算

本文使用「 」及「 」代表「空白字元」與□Enter 換行字元」，並且將使用者輸入的部份使用灰階方式顯示。另外，題目的執行結果中，如果出現「(」、「)」、「:」、「;」、「.」與「,」等符號，皆為英文半形！

P1 五子棋：棋盤檢查

五子棋是一項非常有趣的棋奕遊戲，由分持白子與黑子的雙方在縱橫19道的棋盤上輪流落子。在正式規則中，黑子□Black□永遠為先手，白子□White□為後手。雙方必須輪流落子，且每次只能落下一枚棋子。

根據這個基本的先後手規則，一個合法的棋盤狀態（尚未結束或剛落子）必須符合以下邏輯：

- **當輪到黑子落子時：**□
  - 棋盤上的黑子數量□B□與白子數量□W□必須相等□$B = W$□□
- **當輪到白子落子時：**□
  - 棋盤上的黑子數量□B□必須比白子數量□W□多一個□$B = W + 1$□□
- 在棋盤中：
  - 字元'**1**' 代表黑子。
  - 字元'**0**' 代表白子。
  - 字元'**.**' 代表未落子處。

```
[3:23 user@ws p1] cat▲testfile_1.txt↵
Black↵
.▲.▲.▲.▲.▲.▲.▲.▲.▲.▲.▲.▲1▲.▲.▲.▲.▲.↵
.▲.▲.▲.▲.▲.▲1▲0▲.▲.▲.▲.▲.▲0▲.▲.▲.▲.▲.↵
.▲.▲.▲.▲1▲0▲.▲1▲.▲.▲.▲.▲.▲0▲.▲.▲.▲.▲.↵
.▲.▲.▲.▲1▲.▲.▲1▲0▲.▲.▲.▲0▲0▲0▲1▲.▲.▲.↵
.▲.▲.▲.▲0▲.▲.▲1▲.▲1▲.▲.▲.▲0▲.▲.▲.▲.▲.↵
.▲.▲.▲.▲.▲.▲.▲1▲1▲.▲1▲0▲.▲1▲.▲.▲.▲.▲.↵
.▲.▲.▲.▲.▲.▲.▲0▲.▲.▲1▲0▲0▲.▲1▲.▲.▲.▲.↵
.▲.▲.▲.▲.▲.▲1▲.▲0▲.▲0▲1▲0▲0▲.▲.▲.▲.▲.↵
.▲1▲.▲.▲.▲1▲.▲.▲.▲0▲1▲1▲0▲1▲.▲.▲.▲.▲.↵
.▲1▲.▲.▲0▲.▲.▲.▲0▲1▲1▲0▲1▲.▲.▲1▲.▲.▲.↵
.▲1▲.▲1▲.▲.▲.▲1▲1▲1▲1▲.▲.▲.▲.▲.▲1▲.▲.↵
.▲1▲1▲.▲.▲.▲.▲0▲.▲0▲.▲.▲.▲.▲.▲.▲.▲1▲.↵
.▲1▲.▲.▲.▲0▲1▲1▲1▲.▲.▲.▲.▲.▲.▲.▲.▲.▲0↵
.▲.▲.▲.▲.▲1▲.▲.▲.▲0▲.▲.▲.▲.▲.▲.▲.▲.▲.↵
.▲.▲.▲.▲1▲.▲.▲.▲.▲.▲0▲.▲.▲.▲.▲.▲.▲.▲1↵
.▲.▲.▲0▲1▲.▲.▲.▲.▲.▲.▲0▲.▲.▲1▲0▲0▲0▲0↵
.▲.▲.▲.▲1▲.▲.▲.▲.▲.▲.▲.▲1▲.▲0▲.▲0▲.▲0↵
.▲.▲.▲.▲0▲.▲.▲.▲.▲.▲.▲.▲.▲0▲.▲0▲1▲.▲0↵
.▲.▲.▲.▲1▲.▲.▲.▲.▲.▲.▲.▲1▲.▲0▲0▲1▲0▲1↵

[3:23 user@ws p1]
```

因此同學需完成以下函式的實作：

- **isValidboard()**
  - 傳入 19×19 的棋盤陣列與目前輪到的顏色，負責此棋盤是否合法。

完成後，同學應繳交下列檔案：

- **chess.c**：此文件需包含isValidboard()函式的實作。
- **chess.h**：此文件需包含函式宣告。

請參考以下 main.c, define.h 的內容：

```
#define true 1
#define false 0

typedef enum {Black=49, White=48, None=0} Player;
```

```
#include <stdio.h>
#include <string.h>
#include "chess.h"
```

```
int main()
{
    char chessboard[19][19];
    char color_str[10];
    Player player;
    scanf("%s", color_str);
    for(int i=0;i<19;i++)
        for(int j=0;j<19;j++)
            scanf(" %c",&chessboard[i][j]);

    player = (strcmp(color_str,"Black")==0)?Black:White;
    (isValidboard(chessboard, player) == true) ?
        printf("Valid!\n") :
        printf("Invalid!\n");
}
```

另外本題將使用以下 Makefile 進行編譯：

```
all: main.c chess.o
    gcc main.c chess.o
chess.o: chess.c
    gcc -c chess.c
clean:
    rm -rf *.o *~ *.out
```

本題的執行結果可參考如下：

```
[3:23▲user@ws▲p1]▲./a.out < testfile_1.txt↵
Invalid!↵
[3:23▲user@ws▲p1]▲./a.out < testfile_2.txt↵
Valid!↵
[3:23▲user@ws▲p1]
```

本題測試檔路徑：

- `/home/stu/public/c2026s/c.hw5/p1/testfile_1.txt`
- `/home/stu/public/c2026s/c.hw5/p1/testfile_2.txt`

- 本題相關的程式碼路徑已註明於檔名右側，同學們可以透過路徑複製到自己的家目錄。
- 本題將以上述提供的 Makefile 進行編譯，請同學作答時**“以題目提供的文件為主”**。
- 本題應繳交檔案如下（**define.h**、**main.c** 及 **Makefile** 則不需繳交）：
  - **chess.c**
  - **chess.h**

p2 五子棋-活三

程式演練 22 提供的程式碼提供了完整的程式碼判斷哪一方已經五子連線取得勝利。請參考課本 P13-18 的程式演練 22，修改程式碼增加功能，讓程式能夠找出棋盤中所有活三的位置。所謂「活三」是指已經完成三子連線的（縱、橫或斜向的連續三個相同顏色的棋子），且其前後兩端都沒有棋子。換句話說，這三顆棋子是若己方的可以選擇下子在活三的前後形成四子連線；若是對手的可以阻擋其形成四子連線。

為簡化規則，本題不需找出洞三，也就是連續的四個位置中，除頭尾外中間的某個位置留空，但是另外三子也為同一方的情況，當然洞三的頭尾外側也沒有對手防堵的棋子。

本題不限制 Turnin 所繳交的檔名，但必須繳交 Makefile 檔案並將可執行檔命名為 a.out且 Makefile 會使用到的檔案必須繳交，否則將導致編譯錯誤。

可參考以下 main.c 的內容，讀入棋盤：

```
#include <stdio.h>
#include "chess.h"

int main()
{
    char chessboard[19][19];

    for(int i=0;i<19;i++)
        for(int j=0;j<19;j++)
        scanf(" %c",&chessboard[i][j]);

    isLiveThree(chessboard);
}
```

本題的執行結果可參考如下：

```
[3:23 user@ws p2] ./a.out▲<▲testfile_1.txt↵
LiveThree▲found:▲(L,▲▲6)▲(M,▲▲7)▲(N,▲▲8)↵
[3:23 user@ws p2] ./a.out▲<▲testfile_2.txt↵
Live▲three▲dose▲not▲exist.↵
[3:23 user@ws p2]
```

輸出格式說明

- 首先找到該棋型中位置處於**最上方**且**最左方**的第一顆棋子（如右圖紅圈處），作為起點。
- 根據紅圈延伸出的**箭頭方向**，依序掃描並輸出棋子座標。
- 當起點為同一棋子時，輸出依序遵循：橫向、縱向、斜右下$\searrow$及斜左下$\swarrow$之輸出順序。
- 輸出順序為先看 **數字 (1~19)**，再看 **字母 (A~S)**
- 若找不到活三則顯示 `Live▲three▲dose▲not▲exist.↵`

因此，右圖範例的正確輸出格式應為：

```
[3:23 user@ws p2] ./a.out▲<▲testfile_3.txt↵
LiveThree▲found:▲(F,▲▲9)▲(G,▲▲9)▲(H,▲▲9)↵
LiveThree▲found:▲(F,▲▲9)▲(E,▲10)▲(D,▲11)↵
LiveThree▲found:▲(J,▲10)▲(K,▲10)▲(L,▲10)↵
LiveThree▲found:▲(J,▲10)▲(J,▲11)▲(J,▲12)↵
LiveThree▲found:▲(J,▲10)▲(K,▲11)▲(L,▲12)↵
LiveThree▲found:▲(D,▲10)▲(D,▲11)▲(D,▲12)↵
[3:23 user@ws p2]
```

本題測試檔路徑：

- /home/stu/public/c2026s/c.hw5/p2/testfile_1.txt
- /home/stu/public/c2026s/c.hw5/p2/testfile_2.txt
- /home/stu/public/c2026s/c.hw5/p2/testfile_3.txt

1. 本題不限制 Turnin 所繳交的檔名，但必須繳交 Makefile 檔案並將可執行檔命名為 a.out
2. Makefile 會使用到的檔案必須繳交，否則將導致編譯錯誤。

- 本題批改時將採**人工批改**
- 請勿使用**AI撰寫**，若是助教發現疑似使用AI撰寫，將會約談同學。
- 約談過程中，若對本人繳交之代碼內容回答模糊、邏輯不通，將以 **0 分**計算。

---

p3 五子棋-死四

承上題並參考程式演練 22 提供的程式碼，修改程式碼增加功能，讓程式能夠找出棋盤中所有死四的位置。所謂「死四」是指已經完成四子連線的（縱、橫或斜向的連續四個相同顏色的棋子），且其前後有一端有對手的棋子防堵。換句話說，這四顆棋子是若己方的可以選擇下子在死四缺口形成五子連線並取得該回合勝利；若是對手的可以阻擋其形成五子連線。

同樣地，本題不需找出洞死四，也就是在連續的四個位置中，除頭尾外中間某個位置留空，但另外三子為同一方棋子的狀況，當然洞死四的頭尾已經有一端被對手防堵的棋子。

本題不限制 Turnin 所繳交的檔名，但必須繳交 Makefile 檔案並將可執行檔命名為 a.out且 Makefile 會使用到的檔案必須繳交，否則將導致編譯錯誤。

可參考以下 main.c 的內容，讀入棋盤：

```
#include <stdio.h>
```

```
#include "chess.h"

int main()
{
    char chessboard[19][19];

    for(int i=0;i<19;i++)
        for(int j=0;j<19;j++)
        scanf(" %c",&chessboard[i][j]);

    isDeadFour(chessboard);
}
```

本題可參考以下的執行結果：

```
[3:23 user@ws p3] ./a.out␣<␣testfile_1.txt↵
Dead␣Four␣found:␣(H,␣11)␣(H,␣12)␣(H,␣13)␣(H,␣14)↵
Dead␣Four␣found:␣(I,␣12)␣(J,␣12)␣(K,␣12)␣(L,␣12)↵
[3:23 user@ws p3] ./a.out␣<␣testfile_2.txt↵
Dead␣Four␣dose␣not␣exist.↵
[3:23 user@ws p3] ./a.out␣<␣testfile_3.txt↵
Dead␣Four␣found:␣(L,␣␣7)␣(K,␣␣8)␣(J,␣␣9)␣(I,␣10)↵
[3:23 user@ws p3]
```

本題測試檔路徑：

- `/home/stu/public/c2026s/c.hw5/p3/testfile_1.txt`
- `/home/stu/public/c2026s/c.hw5/p3/testfile_2.txt`
- `/home/stu/public/c2026s/c.hw5/p3/testfile_3.txt`

1. 本題不限制 Turnin 所繳交的檔名，但必須繳交 Makefile 檔案並將可執行檔命名為 a.out
2. Makefile 會使用到的檔案必須繳交，否則將導致編譯錯誤。
3. 輸出參照p2的**輸出格式**
    1. 若找不到死四則顯示 `Dead␣Four␣dose␣not␣exist.↵`

- 本題批改時將採**人工批改**
- 請勿使用**AI撰寫**，若是助教發現疑似使用AI撰寫，將會約談同學。
- 約談過程中，若對本人繳交之代碼內容回答模糊、邏輯不通，將以**0分**計算。

p4 數獨檢查

數獨（すうどく□Sudoku□是一種數字邏輯遊戲，其名稱的意義是「每一格只有一個數字」。玩家需要 將數字 1~9 填入 9*9 的方格中，在這 9*9 的方格中還需要分成 9 個大宮格（請參考右圖較粗的黑線劃分出井字的區塊產生的九個大宮格， 維基百科-數獨），規則如下：

1. 每一列（水平方向）必須包含 1~9 且不能留空也不能重複
2. 每一行（垂直方向）必須包含 1~9 且不能留空也不能重複
3. 每一個大宮格必須包含 1~9 且不能留空也不能重複

在本題中，測試檔將提供一些完整的 9*9 正確格式（每一格都是 1~9 間其中一個數字）數獨棋盤，在這些數獨棋盤中將使用空白作為每一行的數字分隔；使用換行作為每一列的分隔。請參考以下棋盤範例：

```
[3:23 user@ws p4] cat▲testfile_1.txt↵
5▲3▲4▲6▲7▲8▲9▲1▲2↵
6▲7▲2▲1▲9▲5▲3▲4▲8↵
1▲9▲8▲3▲4▲2▲5▲6▲7↵
8▲5▲9▲7▲6▲1▲4▲2▲3↵
4▲2▲6▲8▲5▲3▲7▲9▲1↵
7▲1▲3▲9▲2▲4▲8▲5▲6↵
9▲6▲1▲5▲3▲7▲2▲8▲4↵
2▲8▲7▲4▲1▲9▲6▲3▲5↵
3▲4▲5▲2▲8▲6▲1▲7▲9↵
[3:23 user@ws p4]
```

請參考下列 main.c 的程式碼:

```
#include <stdio.h>
#include "Sudoku.h"
#define true 1
#define false 0
int main(){
    int board[9][9];

    for(int i=0; i<9; i++){
        for(int j=0; j<9; j++){
            scanf(" %d", &board[i][j]);
        }
    }

    (isValidSudoku(board) == true) ?
        printf("Valid\n") :
        printf("Invalid!\n");
}
```

請觀察主程式，同學必須完成 isValidSudoku() 函式，判斷主程式取回的數獨棋盤是否合乎上述數獨規則。同學需完成並繳交 Sudoku.c 與 Sudoku.h 的 C 語言程式，其中**分別包含** isValidSudoku() 函式的**實作**（**Implementation**）與其 **原型**（**Prototype**）宣告。

本題的相關程式將使用以下的Makefile進行編譯：

```
all: main.c Sudoku.o
    gcc main.c Sudoku.o
Sudoku.o: Sudoku.c Sudoku.h
    gcc -c Sudoku.c
clean:
    rm -f *.o *~ a.out
```

本題可參考以下的執行結果：

```
[3:23 user@ws p4] ./a.out▲<▲testfile_1.txt↵
Valid↵
[3:23 user@ws p4] ./a.out▲<▲testfile_2.txt↵
Invalid!↵
[3:23 user@ws p4]
```

本題測試檔路徑：

- /home/stu/public/c2026s/c.hw5/p4/testfile_1.txt
- /home/stu/public/c2026s/c.hw5/p4/testfile_2.txt

本題應繳交 Sudoku.c 與 Sudoku.h 兩個檔案，至於 main.c 與 Makefile 則不需繳交。