TEAM 11

Basic Skill

- **請解釋以下編譯指令中，每一個參數代表的意涵。**
  g++-8 -std=c++17 -O2 -Wall -Wextra -fPIC -I./ -shared AITemplate/Porting.cpp -o ./build/a1.so

  g++-8 我們改成 g++，要 compile 後面的檔案。

  -std=c++17 是指啟用 C++17 這個標準完整的功能。

  -O2 為設定 optimized 的 level，內容是會犧牲部分編譯的速度，但是在跑 code 上會幾乎使用所有可以用來優化的算法。

  -Wall 是指編譯後會把 warning 的部分都顯示出來，但是有部分是顯示不出來的，這個時候就要靠-Wextra 把剩下的都顯示出來。

  PIC 是指 position independent code，不加上 fPIC 的話，加載.so 檔需要重新定位，所以 fPCI 的意思就是要編成 position independent code，這樣不同 process 載入 shared library 時，library 的程式跟資料才能放到記憶體不同位置。

  -I./ adds include directory of header files

  -shared 就是產生一個可以拿來與其他 obecjts 連結的 shared object。

  AITemplate/Porting.cpp -o ./build/a1.so

  -o 代表將前面 build 起來的結果放到後面的 output file。

- **請解釋 Game.h 裡面 call 函數的功能。**

  主要目的是透過這個 function，來 async 的 call AI 所提供的 API，
  兩個 call 是為了因為 AIInterface 內提供的 function 有可能有 return type 或是 void，所以分成這兩種不同的 call。

  第一種是 void call：
  std::invoke_result_t 會 return 根據傳入的函數及參數，實際這個函數回傳的 type。
  std::is_void 得到 bool value，判斷傳入參數是否為 void。
  Std::enable_if_t 當傳入的參數為 true 時，typedef 後面的 type T(int)
  所以整體就是判斷這個 function 是不是 return void，如果為 true，則會進入前面這個 call function，而 typename… Args 則可以將參數壓縮成一個，就可以符合不同的 AI function 所需參數長度。

另一種則是有 return type 的 call：
主要只差在 value == false，也就是當這個 function 不是 return type 不為 void，則透過 auto 以及 invoke_result_t 來決定 call 的 return type。

最後闡述其內容，就是透過 async 來同步執行另一個要 call 的 AI function，並且最多等待 1000ms，如果 status 在一秒後仍然不是 ready，則代表超過時間違反規則，直接 exit(-1)

- **請解釋什麼是 Shared Library，為何需要 Shared Library，在 Windows 系統上有類似的東西嗎？**

Shared Library 是動態連結函式庫，與靜態庫不同，動態庫裡面的函式不是執行程式本身的一部分，而是根據執行需要按需要載入，其執行程式碼可以同時在多個程式中共享。

需要 Shared Library 的原因來自其減少程式大小、節省空間，提高效率、增加程式的可擴充套件性、便於模組化管理等優點。唯一有個缺點就是行為變得複雜許多。

在 Windows 系統上的動態連結函式庫是 Dynamic Link Library(DLL)，一樣能夠實現 Windows 應用程式共享資源、節省記憶體空間、提高使用效率等優點。

Final Project

■ AI Algorithm

先手：

https://www.youtube.com/watch?v=weC1pAeh2Do

https://www.joachim-breitner.de/blog/604-Ultimate_Tic_Tac_Toe_is_always_won_by_X

根據這兩個網頁闡述的內容，我們透過嘗試實現其中的內容，來達到先手必勝的 Algorithm。

先定義 phase 代表不同階段，根據每個 phase 有的可能性來實現不同的行為，主要想法是，因為是先手，所以不斷限制對方，一開始下在中間，之後無論對方下在哪裡，都下在對應 board 的中間，也就是對方只能不斷將中間填滿，雖然將中間放棄送給對方，但我們也幾乎獲得了其他所有 board 的中間。

而當最後中間已經滿了時，則不將其再送回中間，因為這樣對方就可以隨便下，沒有被限制，所以透過下在對應格中，同時進入 phase 2，保持一樣的想法，將對方不斷送回同一格，但我們會獲得其他 board 的對應地方。

最後結果將會是犧牲兩格，但是透過其他 7 格來達到連線。

雖然人為描述並不難懂，但透過程式實現卻很難，最後發現因為可以直接定義好不同 phase 的行為，而不用將規則寫得太 ganeral，因此透過這樣的判斷則能在有定義的棋步內獲得勝利。

後手：

在後手 AI Algorithm 的部分，我們主要使用 DFS 的方法，每一層都是一個 recurtion，對於接下來的每一步評分後累加回來，選擇分數最高的 pair 去回傳。大致上可以分成兩個部分：我們下的回合和敵人下的回合。在演算法中，每加深一層 level 便攻守交換。利用 Min-MAX 搜尋法，我們的回合取最高分，敵人的回合取最低分，來找出最佳的策略。我們的設定的檢驗規則有：我們若能連線得分，則加分；敵人若能連線，則扣分；Board 下的位置剛好對應 Mainboard 的相對位置，則加分，且因為我們認為整個 Mainboard 中間那一格是最重要的，再來是角落，而邊邊是比較不重要，所以若是下在 Board 裡的中間位置也會扣分，而下在邊邊會加分。經過測試，我們發現 level 的層數也會影響我們演算法的勝率，最後我們取 level = 3，即演算法判斷後三手的資訊來評分。

■ 分工

史康一：後手演算法設計、遊戲勝負判定

曹家誠：先手必勝演算法實現、random_AI、Game 流程、GUIInterface

張仲穎：後手演算法設計、遊戲勝負判定

余瑾欣：參與後手演算法設計討論

■ **進度規劃**

Trace code -> Implement basic game playOneRound -> Implement random AI -> JudgeWinState -> firsthandAI(predefined) -> backhandAI -> GUI

■ **心得**

106061214 張仲穎

這次主要負責的部分是在 AI 後手演算法，以及獲勝條件的判別。獲勝條件的判別是有規則可循的，所以相對而言比較簡單一些，然而在後手演算法上，因為有先手必勝的演算法，所以我們僅能盡量去利用 DFS 去累加目前這一步以及後面幾步的分數，然後最後由這個分數作為對於這一步棋的評估。然而分數的標準似乎影響很大，很難說在哪一種條件下一定要獲得多少分、減多少分，我們想了很多自己的 grammar，然後反覆的調整每一項規則的配分，才有了現在這個後手演算法。整體的開發過程還是相當有趣的，從和組員的合作、討論、互動等各個層面上，都讓我學到了很多東西，也交到了新朋友，謝謝我們團隊的所有人！

106072243 余瑾欣

在這一次開發 Ultra Tic-tac-toc 的過程中，我們將演算法分為先手和後手，我參與了後手部分的開發與討論。在我們使用的 Min-max search 中，需要盡量對每一種可能做出評分，而評分的部分我認為是最難的。不像一般 AI 開發可以吃大量的棋譜來建立評分的資料庫，我們需要自己設定怎樣的走法會是高分、怎樣的走法會是低分，以及放在角、邊、中間的權重配置也是由我們去完成。在實作過程中，我們對於評分的方式修改多次，但在最後仍未達成滿意的情況，真的很可惜。在 GUI 設計的部分，我想使用 **SFML** 的 library 來做設計，在 visual studio 上面跑都能夠完成，但下載到 Linux 後卻有 error，且找不出原因，最終只好放棄這個部分，不然原本贏家會有皇室戰爭的背景音樂的。

最後，我很謝謝我們的團隊成員，除了我以外的人都是大神，但又不會嫌我太笨或腦袋跟不上他們的思考，我不懂的地方會跟我說，也願意讓我一起參與他們的討論，謝謝你們！

106060016 史康一

這次 project 跟 tic-tac-toe 很像但是卻是他的進階版，在我原本就會的基礎上卻有了新奇的感覺。我們彼此之間都是試玩過很多次，這個遊戲比原本的 tic-tac-toe 好玩多了，我們一邊比拼腦力一邊嘗試抓出可以產生優勢的下法，但是卻抓不太出來一個方便運行的規則，一直到我們上網查了之後發現了一個先手必勝法的發想，然後研究了一下把影片中沒講清楚的確保，而後手的話應該怎麼下就考倒我們了。一開始也是打算放棄兩大格並以此確保七大格，但是後來發現若是要贏必須先手犯了兩個錯誤，我們就放棄了這個想法，打算給所下的每一

步棋評分，並在最後去下得分最高的那一步，但是如何打分又是一個很嚴重的問題，我們不知道應該如何去。最後我發現這個做法還是有很多地方可以改進，可惜因為時間的原因有點來不及，不過也透過了這次的 final project，我能更好的把近期所學的用上。

最後我也必須謝謝我的組員，因為組員間彼此討論給予意見，我也從我的組員那邊得到很多寶貴的意見，當彼此討論如應該如何實現這樣的功能時，都能從組員那邊聽到有更簡潔的打法，除此之外還有打 code 時的思路想法，組員傳的一些參考資料也對我在完成我的部分時有非常大的幫助，相信以後照著這樣的思路去完成題目一定會更容易。

106060004 曹家誠

這次的作業一開始要先讀懂助教提供的 template，而內容有許多是以往沒看過的，從 linux 的 shared library 到 makefile 中，都有許多是可以學習的。

一開始先是透過 wsl 這個功能來進行開發，接著觀察整個使用 shared library 的流程，都是以往沒看過的。

另外像是 GUIInterface 中提供的 terminal 版本的實作方式，也是以往不曾看過的，舉例來說像是調整游標位置，到自己透過網路查找的改變顏色等等。

最後開發 AI 的過程，整體而言比起之前的作業會更有趣，也更有成就感，相比起來甚至也更好 debug，了解整體架構後甚至也比以往作業更為簡單。

我覺得這次的作業重要的不是實際寫了些什麼，而是透過內容以及網路學習了更多以前沒聽過的名詞以及其內容，才是最大的收穫。

Bonus

- **-Werror -Wextra -pedantic-errors**
  Makefile 中加入以上 flag 仍然可以透過編譯，也就是不會有 warning，一般來說移除 unused variable 就可以了。
  注：環境為 WSL2、vscode wsl extension、ubuntu20.04(LTS)、
  g++ (Ubuntu 9.3.0-10ubuntu2) 9.3.0
- **GUI**
  新增了另一個 UltraBoard 來更清楚的表示每個 board 的輸贏，並且根據不同 Tag 使用不同的顏色，有更明顯的區別。