誰是牛頭王 Take 6!

程式設計(二)-Project

Due to 06/29 AM 00:05／授課老師：紀博文

一、基本資料

姓名：林育辰
系級：資工 111
學號：40771131H

二、檔案有哪些？

- **40771131H_project.c**
- **40771131H_contest.h**
- **contest.c**
- **Makefile**
- **README.pdf**

◎檔案中皆有註解！
◎40771131H_contest.h 提供助教 include 進 contest.c
◎40771131H_project.c 為個人的遊戲檔案

三、如何執行？

請輸入 make
→編譯 40771131H_project.c
→產生 40771131H_project
→執行 $./40771131H_project
按照介面操作即可進行遊戲

四、規則

一、普通規則

1) 手牌依據 project.pdf 規定為 104 張牌 (非說明書的 人數 $\times 10 + 4$)，數值範圍為 1-104(唯一)
2) 每位玩家會拿到十張手牌，十個回合所有玩家手牌出完遊戲結束
3) 每張牌依據其數字擁有其牛頭數：
   - 牛頭數 7：55
   - 牛頭數 5：11, 22, 33, 44, 66, 77, 88, 99
   - 牛頭數 3：10, 20, 30, 40, 50, 60, 70, 80, 90, 100
   - 牛頭數 2：5, 15, 25, 35, 45, 65, 75, 85, 95
   - 其餘數值為牛頭數 1
4) 牌面共四排，每排至多可放 5 張
5) 起初牌面會有 4 張排頭，各玩家挑選一張手牌覆蓋，待所有玩家出完，同時翻開牌面，並由數字小至大放入各排內，牌會依據其數字，放入「各排最後一張最接近的數字」，若比任意一排數字還要小，則玩家必須選擇一排，將整排的牌拿走，該玩家所出的排當作該排的新排頭
6) 若放置牌時，該排已經有 5 張牌，則該玩家必須拿走整排，並以該玩家所出的牌作為新排頭
7) 最終依牛頭數由「少至多」排名

二、變體規則

因在線上遊玩該款遊戲，有時候一個不小心就會變成真正的「牛頭王」，因此思考，也許遊戲勝利規則改為「拿最多牛頭數」勝利，在過程中的策略就會幾乎相反，可是這樣只是單純的修改遊戲(反過來而已)。

如果說，設定拿指定的牛頭數以上勝利，若未超過則按照原本的規則計算勝利狀態，那麼，玩到一半就會有人「走偏」開始往吃牌路線前進，他有他的策略，而一些人則按照「儘量少吃牌」路線前進，由於兩種不同策略的考量，遊戲的牌面將會更難預測，亦可大大增進不確定性以及刺激程度。

因此，在此規則，原則上所有規則同「普通規則」，在計算排名時，假使指定為x，而某玩家其牛頭數 $\geq x$，則按照牛頭數多至少排名，若玩家未符合此條件，則按照牛頭少至多排名，此外，為符合「誰是牛頭王」的名稱，只要得了狂牛症則排名會在未得狂牛症的玩家前面。

即在本規則中，大於等於 x 玩家稱為「狂牛症」，吃越少則狂牛症指數越低。

→拿超多牛頭數，拿很少牛頭數都有可能獲勝。

——example——

狂牛症變身條件：大於等於 30 (自行指定)

A 玩家：牛頭數 35 → 勝利指數： $35 - 30 = 5$ [狂牛症]

B 玩家：牛頭數 20 → 勝利指數：20

C 玩家：牛頭數 5 → 勝利指數：5

D 玩家：牛頭數 12 → 勝利指數：12

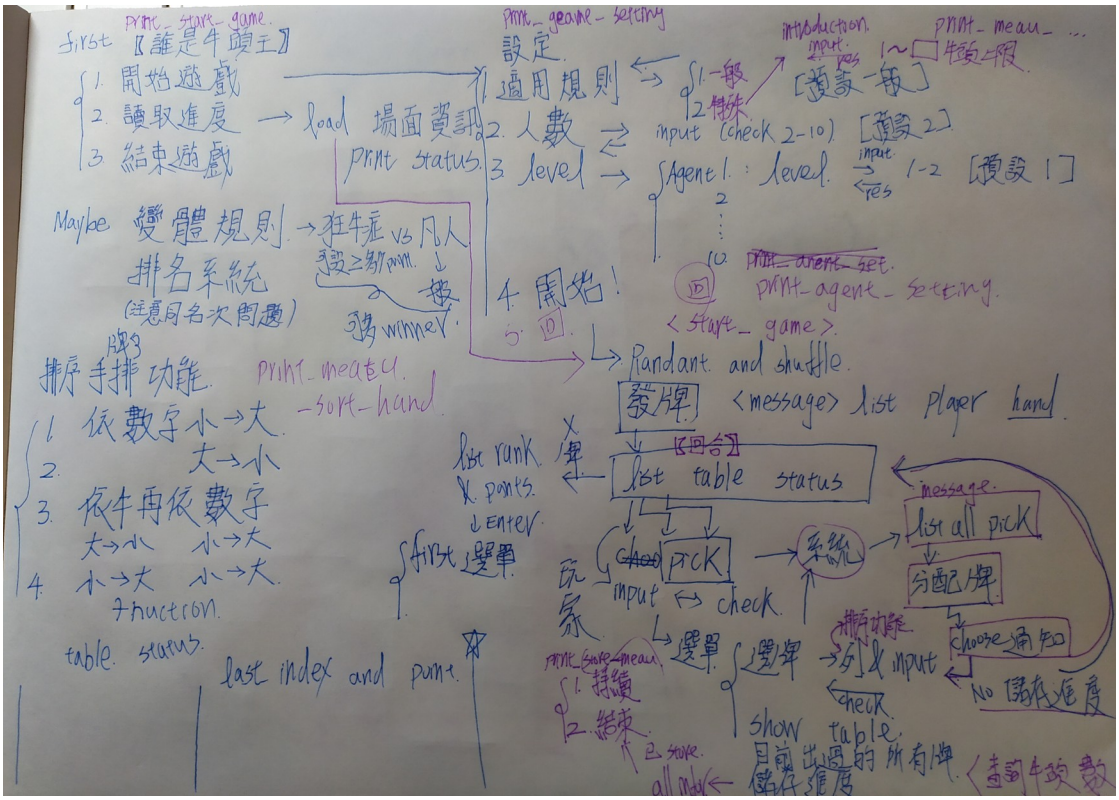
則排名： **A(5) = C(5) > D(12) > B(20)**

p.s 若想要線上嘗試 take 6!，可以在下列網址遊玩：

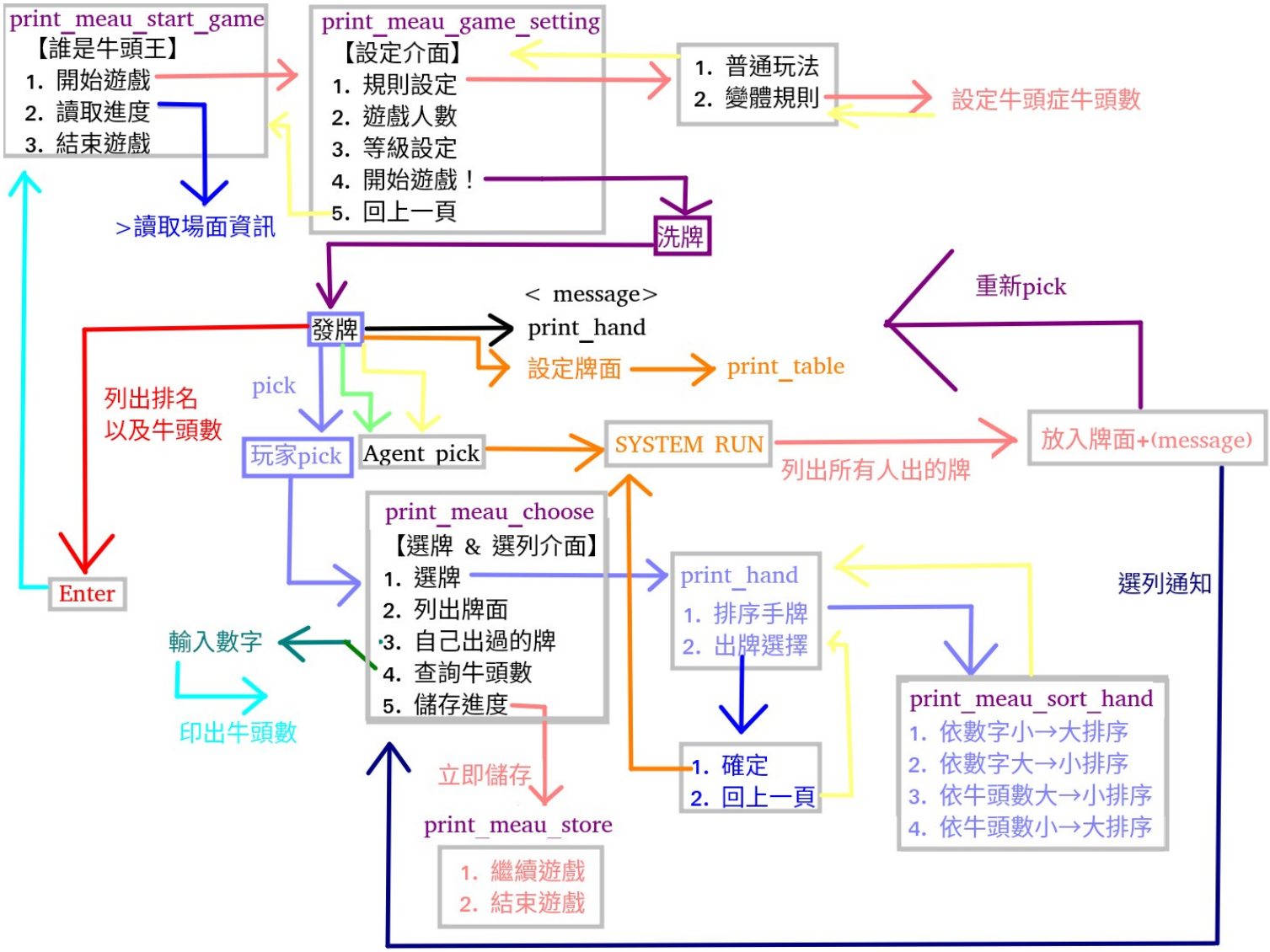
URL: https://boardgamearena.com/lobby

五、使用者介面設計

(一) 草圖



(二) 正式流程圖


print_meau_start_game
【誰是牛頭王】
1. 開始遊戲
2. 讀取進度
3. 結束遊戲
>讀取場面資訊
print_meau_game_setting
【設定介面】
1. 規則設定
2. 遊戲人數
3. 等級設定
4. 開始遊戲！
5. 回上一頁
1. 普通玩法
2. 變體規則
設定牛頭症牛頭數
洗牌
發牌
< message>
print_hand
設定牌面
print_table
重新pick
列出排名
以及牛頭數
pick
玩家pick
Agent pick
SYSTEM RUN
列出所有人出的牌
放入牌面+(message)
Enter
print_meau_choose
【選牌 & 選列介面】
1. 選牌
2. 列出牌面
3. 自己出過的牌
4. 查詢牛頭數
5. 儲存進度
print_hand
1. 排序手牌
2. 出牌選擇
選列通知
輸入數字
印出牛頭數
1. 確定
2. 回上一頁
print_meau_sort_hand
1. 依數字小→大排序
2. 依數字大→小排序
3. 依牛頭數大→小排序
4. 依牛頭數小→大排序
立即儲存
print_meau_store
1. 繼續遊戲
2. 結束遊戲

六、函式設計

主要有以下的函式：

P.s. 為 contest 限定的幾個函式內容
void CSIE_40771131H_setup(int32_t id);
void CSIE_40771131H_deal(const int32_t cards[10]);
int32_t CSIE_40771131H_pick(const int32_t table[4][5], const int32_t score[2], const
int32_t last[8], const int32_t card[10]);
int32_t CSIE_40771131H_choose(const int32_t table[4][5], int32_t card);
project 的函式內容
int32_t CSIE_40771131H_transform(int32_t card);
input：牌的數值 output：該數值所代表的牛頭數 特例：數值 0 以及-1 輸出為 0
void CSIE_40771131H_determine_last_index_index(const int32_t table[4][5], int32_t last_index[4], int32_t score[4]);
input：牌面、(空)牌面中每排最後一個的位置、(空)每排的牛頭數總和 →從給定牌面，計算後兩個的數值並存入
void __setup_shuffle(int32_t card[], int32_t n);
input：全部卡的陣列、陣列大小 →配置陣列內容為 index+1
void range_shuffle(int32_t card[], int32_t n);
input：全部卡的陣列、陣列大小 →從 n-1 到 1，每次都隨機取一個 index 和第 i 項交換 →洗牌
void print_player_hand(int32_t player, int32_t **hand_card, int32_t n);
input：玩家序號、所有手牌陣列、手牌數 →印出指定玩家手牌以及相對應的牛頭數
void print_table_status(int32_t table[4][5], int32_t row, int32_t col);
input：牌面、列數、行數 →印出牌面的點數、牛頭數，每列尾巴列出總牛頭數、排尾位置

int32_t **declared_two_pointer(int32_t row, int32_t col);
input：列數、行數 →用雙重指標建造一個二維空間 output：該二維空間的指標位置
int32_t look_up_hand_index(int32_t card[10], int32_t val);
input：手牌、數值 output：該數值在手牌中的位置，沒找到則回傳-1
int32_t Agent_pick_level_1(int32_t table[4][5], int32_t card[10]);
input：牌面、手牌 output：第一級的電腦決策後選擇的牌(數值)
int32_t Agent_choose_level_1(int32_t table[4][5], int32_t card);
input：牌面、牌的數值 output：第一級的電腦決策後選擇要吃掉的列
int32_t Agent_pick_level_2(int32_t table[4][5], int32_t card[10]);
input：牌面、手牌 output：第二級的電腦決策後選擇的牌(數值)
int32_t Agent_choose_level_2(int32_t table[4][5], int32_t card);
input：牌面、牌的數值 output：第二級的電腦決策後選擇要吃掉的列
void pick_one_row(int32_t table[4][5], int32_t score[], int32_t player, int32_t row);
input：牌面、玩家點數、玩家序號、列 →讓指定玩家吃掉指定的列(牛頭數增加)、清除該列
int32_t put_one_card(int32_t table[4][5], int32_t score[], int32_t player, int32_t card[10], int32_t val);
input：牌面、玩家點數、玩家序號、手牌、數值 →根據數值找到「該放的列」，若小於所有列的數值則回傳-1 →若該列已有五張牌，則以使用者的牌作為排頭，並讓該使用者吃掉整排 →其餘則直接加一張牌在列尾巴 output：回傳應該放的「列」，若無則回傳-1
int32_t put_one_card_and_clear_hand(int32_t card[10], int32_t val);
input：手牌、數值

→從手牌中找到指定數值，並清除掉 output：若找不到數值則回傳-1，成功輸出 1
int32_t put_one_card_in_table_row(int32_t table[4][5], int32_t player, int32_t card[10], int32_t val, int32_t row, int32_t table_last[]);
input：牌面、玩家序號、手牌、數值、列序號、牌面每列最後的 index →將牌放置到指定的列尾巴，並從玩家手牌中清除此牌 output：成功回傳 1，失敗回傳-1
void print_meau_start_game(); **void print_meau_game_setting();** **void print_meau_agent_setting();** **void print_meau_store();** **void print_meau_sort_hand();**
→印出各式選單
其它

typedef struct _last_status{
int32_t last;
int32_t row;
}**last_status;**
(結構為 index 和所在的列序號)
 	
int32_t compare(const void *a, const void *b){
if (((last_status *) a)->last > ((last_status *)b)->last) return 1;
return 0;
}
(qsort 根據 last 的數值排序)
 	
int32_t CSIE_40771131H_cards[10] = {0};
int32_t CSIE_40771131H_id =0;
(cards 和 id 的紀錄)

其它擴增函式
其它撰寫有關讀檔、存檔和電腦分級的函式，在程式黨內有註解介紹

七、特殊功能

- 存檔、讀檔功能
- 普通規則、特殊規則可選擇
- 電腦 AI 共三層分級
- 接受 2-10 人遊玩

八、CONTEST

- 助教 include 時，應該整份檔案都 include 並且變數名稱不得更改
- 助教必須將 function pointer 指定為 CSIE_[ID]__XXXX 的型式
- CONTEST.C 內容不得有編譯錯誤，請必須 include “40771131H_contest.h”
- 根據助教回信，助教會手動將 struct 的 function pointer 指定為同學的.h 檔內 function，因此若直接執行附檔的 contest.c 會有編譯錯誤，還請助教處理完 contest.c 內容後再行處理，此外務必記得 include 該檔 (40771131H_contest.h)