Etap	1	2	3	4	Suma
Punkty	7	6	3	5	21
Wynik					

L3: Bridge game

Zarys: Trzeba napisać program, który zasymuluje uproszczoną grę w brydża. Rozgrywka będzie bez atu i bez licytacji. Najpierw przy pomocy shared memory zostanie utworzona ręka każdego gracza (rozdawanie) i przy pomocy shared bariery zbierze się 4 graczy. Później rozpocznie się rozgrywka. Zaczyna rozdający pierwszy proces. Każdy proces musi dokładać do koloru. Gdy cala lewa jest na stole, każdy ocenia czy linia bierze czy nie. Wtedy odkłada swoja kartę z oznaczeniem zebrana/oddana. Po 13 lewach każdy z graczy wypisuje na ekran swój wynik. Gdy wszyscy wypiszą swój wynik dealer musi sprzątnąć shmem.

Stages:

1. [7 p.] Zaimplementuj inicjalizacje stołu w pamięci dzielonej (funkcja `table_init`). Proces gracza czeka na 3 innych graczy i się kończy. Pamięć dzielona powinna być inicjalizowana tylko przez pierwszego gracza.

2. [6 p.] Każdy gracz po dołączeniu do stołu powinien otrzymać swoje miejsce przy stole (numer 0,1,2 lub 3). PO zebraniu się 4 graczy przy stole należy prawidłowo zniszczyć obiekt pamięci dzielonej i zwolnić wszystkie zasoby. W tym celu zaimplementuj funkcje `table_destroy` oraz `table_close`. **Podpowiedź:** Bariera po zwolnieniu blokady przekazuje jednemu wybranemu wątkowi specjalną wartość. Wykorzystaj to do wybrania procesu zwalniającego usuwającego obiekt pamięci dzielonej.

3. [3 p.] Po zebraniu wszystkich czterech graczy każdy gracz powinien pobrać 13 kart ze stołu (patrz `table.h` i posortować 13 kart w swojej ręce (patrz `hand.h`). W tym celu zaimplementuj funkcję `start_game`. **Podpowiedź:** Do sortowania wykorzystaj funkcje `qsort` (`man 3p qsort`).

4. [5 p.] Dodaj logikę rozgrywki gracza. Przy dokładaniu kart musisz rozróżnić logikę wistującego (rozpoczynającego) i dokładającego gracza. Po każdej lewie (zrzutce) należy ocenić która para graczy zebrała lewę. W tym celu zaimplementuj funkcje `play_trick` oraz `asses_result`.