專案建構

直接使用 `make` 指令即可，輸出的執行檔叫做 `poker.out`。執行環境應為 Linux（雖然理論上其他平台也要可以動，但其他平台需要先準備好 `make` ）， dependency 只有 GNU make 而已，並無額外依賴。

介面介紹

```
Now operating on player 1
0) Quit
1) Deal n cards to player
2) Print card of specific suit with deal order
3) Print card of specific suit with reverse deal order
4) Print card of specific suit with size order
5) Delete card of specific suit
6) Switch operating player
7) Play game!
Input command:
```

基本上是切換兩個不同使用者的功能是提供在選項 6 上面，選項 7 是加分題，其他的指令都如字面上所述，與老師的作業要求一一對應。

複雜度分析

基本上我實作了一個與 `std::list` 行為相同且提供相同 method 的 double linked list ，位於 `mystl` 底下叫作 `list` （加起來就是 `mystl::list` ）。設計上為了滿足與 C++ 的 STL 相同的 iterator 行為，因此採用前後皆有偽節點的作法。

`mystl::list` 提供了以下幾個 method ：

emplace / emplace_back / emplace_front

這些 method 可以在指定位置插入元素到 linked list 中，時間複雜度是 $\mathcal{O}(1)$ ，因為只需要新造一個節點後接上即可。

erase / pop_back / pop_front

這些 method 可以移除指定的元素，並回傳他的下一項。時間複雜度是 $\mathcal{O}(1)$ 若只有一個要移除的元素。

size / empty

追蹤目前容器內的元素量，簡單的放置了一個計數器去追蹤，回傳計數器的狀態，因此時間複雜度是 $\mathcal{O}(1)$ 。

merge

合併兩個已經排序好的 linked list ，這個動作為線性複雜度。

sort

這是一個 merge sort 的實作，時間複雜度是 $\mathcal{O}(n \log n)$ 。

選擇某一花色，依發牌拿到的順序一一列出 / 依發牌拿到的相反順序一一列出

這兩個功能都是線性複雜度，因為不涉及排序，只有遍歷。

選擇某一花色，依牌色大小由小而大一一列出

這個功能使用了前面描述的 `sort` ，因此時間複雜度是 $\mathcal{O}(n \log n)$ 。

刪除一張牌

這裡透過標準函數庫提供的 `std::upper_bound` 尋找對應要刪除的排，複雜度應為線性。然後再線性複雜度的找到要刪除的元素。

後記

阿雖然寫到遲交 QQ ，不過使用了 Test driven development 的方法進行開發，是個挺有趣的體驗。自造與 STL 相同介面的 `list` 也非常有趣，不過也是遲交的罪魁禍首。