1. 首先，先不用管 CirGate。先寫 CirMgr::readCircuit(). 而 readCircuit() 的動作，照順序應該要包含：開檔、read header, read input def, read output def, read AIG def, read symbols, read comment 等。你可以先宣告一些 private helper functions 來代表這些動作。

2. 針對 1. 的每一個 functions, 直接先用 ifstream::operator >> 來讀取，配合適當的 variables/types, 至少先把沒有 error 的檔案讀得進來，而如果是非 space 相關的 errors 應該也可以抓得到。此時，因為還沒有寫 CirGate, 你可以先把讀進來的變數直接用 cout 印出來 (順便印一下 line number)，配合一些 debugging message, 你至少可以確定你的 parsing 流程是順的。這就是這個作業的 MVP.

3. 先來寫 CirMgr::readHeader() 這個 helper function, 把 header 讀進來，把 MILOA 五個變數存到 CirMgr 裏頭.

4. 再來寫 CirMgr::readInput(). 你可以考慮定義一個 class CirPiGate : public CirGate, 然後 construct an PI gate by CirGate* pi = new CirPiGate(gateID, lineNo), 其中 gateID 就是這個 PI 的 var ID, lineNo 就是 line number. 在 CirMgr 裏頭開一個 _piList 把這個 pi 存進去.

5. 再來寫 CirMgr::readOutput(). 一樣定義一個 class CirPoGate : public CirGate, 不過他的 gate ID 應該要從 CirMgr::M + 1 開始編號, 而 AIG 檔中定義這個 PO 的數字，應該是這個 PO 的第一個 (也是唯一一個) fanin，你應該把這個數字先存到這個 PO 的 _fanin0 (or _faninList[0]) 上面，而不是存到這個 PO 的 gate ID. 至於 inverted phase, 你可以使用講義中講的 CirGateV 來存, 或者是，簡化起見，先用一個 bool _invPhase0 (or _invPhase[0]) 來存起來.

6. CirMgr::readAig() 應該也跟 readPO() 差不多，只不過每個 AIG gate 會有兩個 fanins.

7. 至於 const 0, 你可以考慮定義一個 CirMgr 的 static data member, 不過請記得要在 .cpp 檔裡頭 initialize 這個 static data member.

8. readSymbol() 以及 readComment() 可以最後再做

9. 上面把 PI, PO, AIG 都讀進來，但現在有個問題，就是我怎麼從 gate ID 找到這些 gates? 你可以選擇使用一個 map<unsigned, CirGate*>, 用 gate ID 來找到 Gate*, 或者是直接宣告一個 vector<CirGate*>, 用 gate ID 直接來 index Gate*. Of course, 這個 map or vector 是一個 CirMgr 的 data member (let’s call it _gateList). 你可以把存 _gateList 的動作夾到 4, 5, 6 步驟裡頭.

10. 接下來就是要來把 connection 接起來囉 (CirMgr::connect())！你可以 go through _gateList 裡頭的每一個 gate, 如果是 PI, 則 skip, 如果是 PO or AIG gate, 則把原先存在 _fanins 裡頭的數字 (well, 如何把數字存在 CirGate* 型態的變數上，你可以參考作業四的辦法)，去 look up _gateList, 找到對應的 CirGate*, 然後把它取代這個 _fanin.

11. 理論上，做到這邊，你應該就把電路接好囉！接下來你可以寫一個 CirMgr::genDFSList() 用 post-ordered traversal 把 DFS 的 gates 的結果存到 CirMgr::_dfsList 上面.

12. 把 CirMgr 裡頭的 printSummary(), printNetlist(), printPIs(), printPOs() 寫一寫，這樣你至少可以測看看 CIRRead 以及 CIRPrint 這兩個 commands 有沒有寫對 (note: error 以及 floating 還沒有 handle).

13. 接下來就是 CIRGate, CIRPrint -Floating, 以及 CIRWrite, 這部分就自己寫囉！

14. 最後寫一個 CirMgr::reset(), 讓 CIRRead -Replace 可以 work.

15. 有時間的話，可以再優化你的程式，或者是做多一點 error handling.

16. 這樣大致上是完成囉！是不是沒有很難？

Fighting!