題目1：

執行環境：mac terminal (using vscode)

執行步驟：swipl -q -s problem\_1.pl  
 直接輸入數字即可，ppt上的例子：100.

程式碼說明：

* 檢測是否是質數：  
  如果輸入值小於等於2的話，如果是2回傳true（不等於1），如果是1就回傳false -- @a  
  如果輸入值大於2 -- @b  
  就從2~輸入值開始取輸入值的餘數，若等於零就回傳true（@b取到false），因此可以判斷是否為質數 -- @c
* 建立事實，X是輸入值，Y從2開始找，若Y是質數，則再去看X-Y是不是質數，如果是則輸出，如果不是則讓Y+1繼續去找 -- @e  
  當Y找到自己的時候（X==Y），代表此數找不到兩質數相加則輸出false -- @d
* 輸入值，判斷是否為偶數且要大於3 -- @f
* div(X,Y) :- 0 is X mod Y ; X > Y+1, div(X, Y+1). ## @a
* isPrime(X) :- X=<2, X=\=1, !. ## @b
* isPrime(X) :- X>2, not(div(X, 2)). ## @c
* % X is input, Y is 2 ~ input
* from\_2\_to\_input(X, Y):- X==Y, !, false. ## @d
* from\_2\_to\_input(X, Y):- ## @e
* isPrime(Y), N is X-Y, isPrime(N), write(Y), write(" "), write(N), nl;
* M is Y+1, from\_2\_to\_input(X, M).
* main:- ## @f
* read(X),
* 0 is X mod 2, X > 3, from\_2\_to\_input(X, 2),
* halt;halt.
* :- initialization(main).

題目2：

執行環境：mac terminal (using vscode)

執行步驟：swipl -q -s problem\_2.pl

依照順序輸入，ppt上的例子：6.(enter)1.(enter)2.(enter)2.(enter)3.(enter)1.(enter)4.(enter)4.(enter)5.(enter)4.(enter)6.(enter)3.(enter)3.(enter)4.(enter)5.(enter)6.(enter)1.(enter)2.(enter)

程式碼說明：

* 根據LCA的Pseudocode去寫的
  + if x == y, print x -- @a
  + else if x = parent(y), print x -- @b
  + else LCA(parent(x), y) -- @c
* 但是在找 x = parent(y)，要一路找到最上面，因此先定義事實 -- @d  
  如果上一個並不是parent的話，就找出B的parent，然後再recursive回去找A跟B的parent的共同ancestor -- @e
* 建立輸入幾次的最後一步事實 -- @f  
  用Y當作參數去做出for迴圈，然後再依次數入並新增事實 -- @g @h
* 建立要跑幾次LCA的最後一步事實 -- @i  
  用Y當作參數去做出for迴圈，然後再依次數入並跑LCA -- @j
* /\* LCA Pseudocode
* \* LCA(x, y) =
* \* { if x = y or x = parent(y), return x }
* \* { else LCA(parent(x), y) }
* \*/
* ancestor(A,B) :- parent(A,B). ## @d
* ancestor(A,B) :- parent(X,B), ancestor(A,X). ## @e
* lca(A,B) :-
* A==B, write\_ln(A); ## @a
* ancestor(A,B), write\_ln(A); ## @b
* parent(X,A), lca(X,B). ## @c
* read\_x\_times(1):- true, !. ## @f
* read\_x\_times(X):- ## @g
* Y is X - 1,
* read(N), read(M),
* assert(parent(N, M)), ## @h
* read\_x\_times(Y).
* lca\_x\_times(0):- true, !. ## @i
* lca\_x\_times(X):- ## @j
* Y is X - 1,
* read(N), read(M),
* lca(N, M),
* lca\_x\_times(Y).
* main:-
* read(X),
* read\_x\_times(X),
* read(Y),
* lca\_x\_times(Y),
* halt;halt.
* :- initialization(main).

題目3：

執行環境：mac terminal (using vscode)

執行步驟：swipl -q -s problem\_3.pl

依照順序輸入，ppt上的例子：6.(enter)6.(enter)1.(enter)2.(enter)2.(enter)3.(enter)3.(enter)1.(enter)4.(enter)5.(enter)5.(enter)6.(enter)6.(enter)4.(enter)2.(enter)1.(enter)3.(enter)1.(enter)5.(enter)

程式碼說明：

* 為了利用原library中的reachable函數， -- @a  
  因此必須先將所有edge（定義）放進去ES變數內 -- @b  
  把所有的ES轉成變數G(graph) -- @c  
  之後就可以用reachable函數去找指定node的所有reach到的node並存成list到Path變數內 -- @d  
  之後就查看Path list內是否有指定的node就回傳Yes，否則傳No -- @e
* 建立輸入幾次的最後一步事實 -- @f  
  用Y當作參數去做出for迴圈，只是因為library中是設成單向的graph，因此只要新增雙向的事實就符合題目標準 -- @g @h
* 建立要跑幾次Path的最後一步事實 -- @i  
  用Y當作參數去做出for迴圈，然後再依次數入並跑LCA -- @j
* path(X,Y) :- ## @a
* findall(A-B, edge(A,B), Es), ## @b
* vertices\_edges\_to\_ugraph([],Es,G), ## @c
* reachable(X,G,Path), ## @d
* member(Y, Path), ## @e
* write\_ln("Yes"),!,true;
* write\_ln("No"),false.
* read\_x\_times(0):- true, !. ## @f
* read\_x\_times(X):- ## @g
* Y is X - 1,
* read(N), read(M),
* assert(edge(N, M)), assert(edge(M, N)), ## @h
* read\_x\_times(Y).
* path\_x\_times(0):- true, !. ## @i
* path\_x\_times(X):- ## @j
* Y is X - 1,
* read(N), read(M),
* path(N, M),
* path\_x\_times(Y).
* main:-
* % X is num of nodes, Y is num of edges
* read(X), read(Y),
* read\_x\_times(Y),
* read(Z),
* path\_x\_times(Z),
* halt;halt.
* :- initialization(main).