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).
```