Justificar los efectos que en la resolucion SLD tendría la consideracion de un algoritmo de unificación que no aplicase un occur-check. Ilustralo con un ejemplo.

Test de cidicidad -> occur check X no forma parte de la estructura correspondiente a Y.

Si no se aplica el ocur check el algoritmo de unificacion entraria en un ciclo sin fin.

Ejemplo: Términos igual (x,x) e igual (y,f(y))

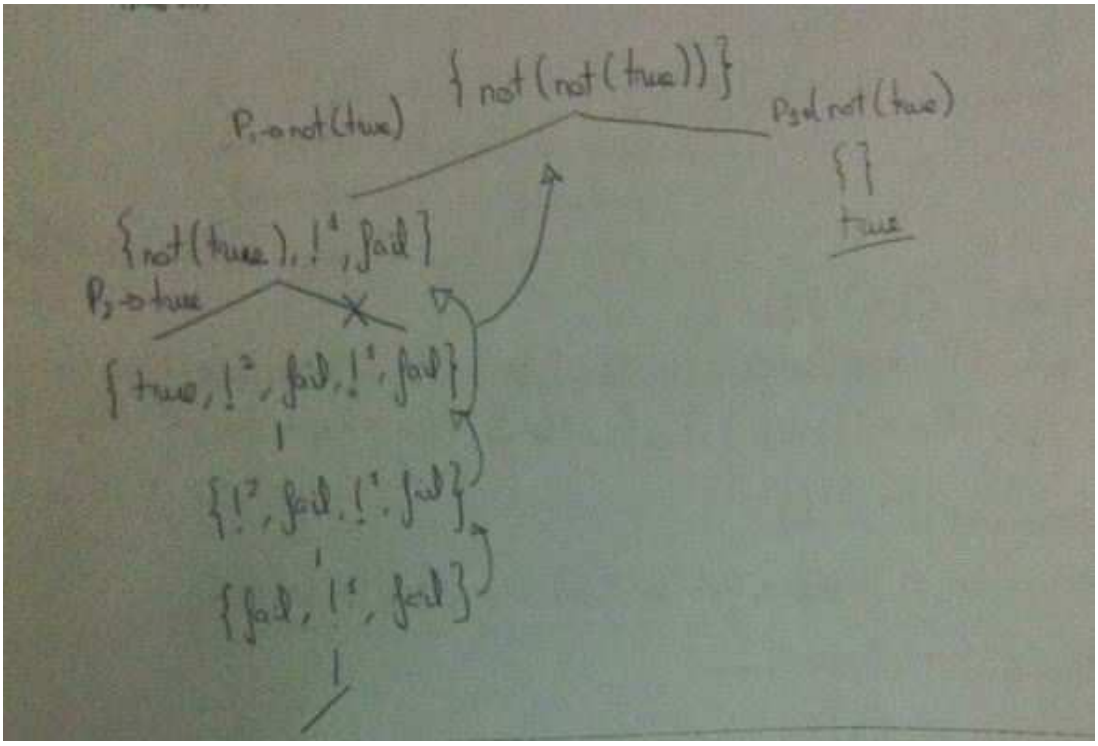
igual (x,x)= igual (y, f(y))
x=y
x=f(y)
y=f(y)

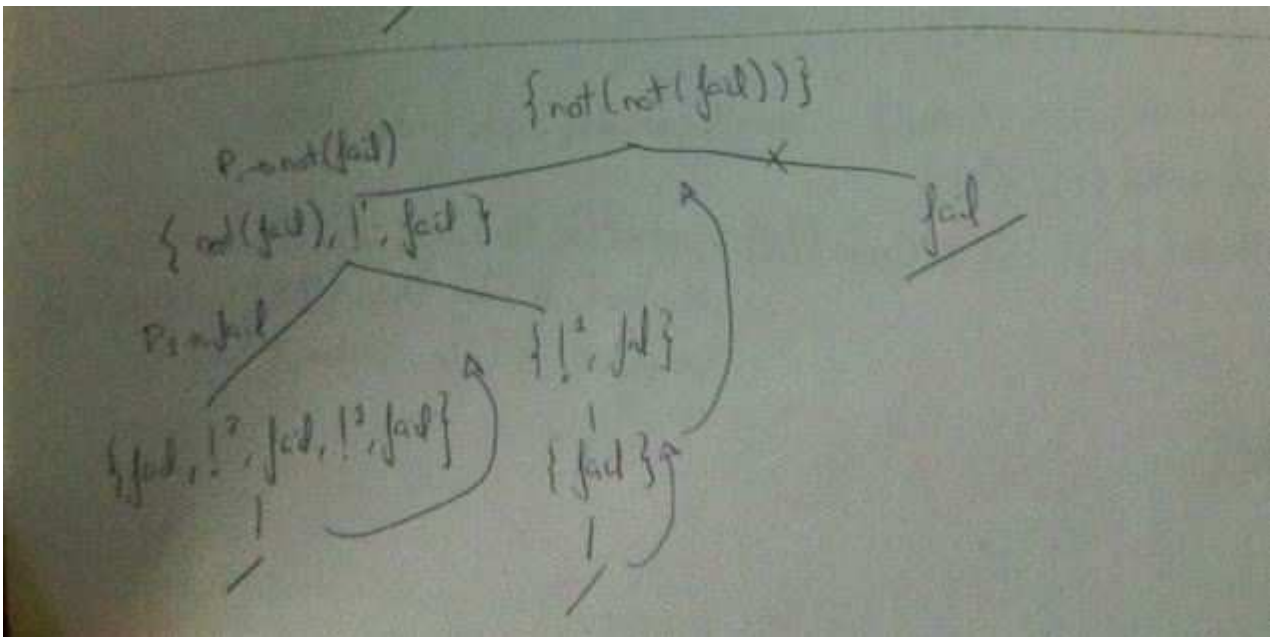
-En la ultima parte aparece f(y), por lo tanto si continuamos con el proceso ocurrira que sustituiremos toda ocurrencia de y por f(y)
x<-- y <-- f(y) <-- f(f(y))....

-Por lo tanto la unificación ha entrado en un ciclo.

Demostrrar que not (not(P))= P. Bastará con probar que not(not(true))= true y que not(not(fail))=fail.

```
not(P):- P,!,fail.
not(P).
```





Razonar la verdad o falsedad en la afirmación siguiente.
"El precidado fail implementa por si solo el concepto de negacion por fallo".

Falso, necesita otros conceptos ademas del fail. Cuando se verifica que P es cierto se deberá colocar un corte, por lo tanto el predicado fail no lo implementa por si solo, quedando el problema rsuelto de esta manera:

```
not(P):- P,!,fail.
not(P).
```

Dado el programa PROLOG siguiente:

igual(x,x).

mostrar y justificar su comportamiento frente a la pregunta :- igual(Y,f(Y)).

Ø = {}	Ø= {}	
igual(X,X)=igual(Y,f(Y))	X=Y	Ø={X<-- Y} fail
	X=f(Y)	Y=f(Y)

- En la última parte aparece f(Y), por lo tanto si continuamos con el proceso ocurrirá que sustituiremos toda ocurrencia de Y por f(Y)
  X<--- Y <--- f(Y) <-- f(f(Y)) <-- f(f(f(Y)))....
  - Por lo tanto la unificación ha entrado en un ciclo. Se puede ver que sin test de ciclicidad entra en un ciclo sin fin.
  -

Explicar cual es la relacion entre la recursividad izquierda en las clausulas y las ramas infinitas en un arbol de resolucion PROLOG.

-La recursividad izuiqerda cause las ramas infinitas porque para la resolucion de la llamada recursiva (de izquierda) es necesario volver a llamar a la misma clausula que generase, requiriendo llamarse a si mismo otra vez infinitas veces para intentar resolverse, generando un ciclo con ramas infinitas.