

```
2) (a b c) (c d)
   (defun union(seta setb)
     (cond ((null seta) setb)
           ((member (car seta) setb) (union (cdr seta) setb))
           (t (cons (car seta) (union (cdr seta) setb)))))
```

Manual trace: call 1: (Union (a b c) (c d))
call 2: (Union (b c) (c d))
call 3: (Union (c) (c d))
call 4: (Union () (c d))

call 4 returns (c d)
call 3 returns (c d)
call 2 returns (b c d)
call 1 returns (a b c d)

- 3) eats(robie, icecream).
eats(robie, burgers).
eats(robie, fries).
eats(tim, veggies).
eats(tim, juice).
eats(tim, icecream).

```
~/Desktop — guili002@empress:~/cs351 — ssh guili002@empress.csusm.edu — 130x40
[guili002@empress cs351]$ bp
BinProlog, #12.00 Copyright (C) Paul Tarau 1992-2012.
Open-sourced under GPL v.3 licence at
http://www.gnu.org/licenses/gpl-3.0.txt.
(C-ified standalone)
(with heap GC enabled) (64 bits)
Detected hostname: undetected (type bp -p10 to detect host)
Start loading system code from <internal code array>.
Finished loading system code.
Started Prolog Runtime System 1.
[?- [food].
compiling(to(mem),food.pl,...)
bytes_used(code(448),strings(52),symbols(64),htable(360),total(924))
compile_time(0)
[?- eats(tim, X), eats(robie, X).
[X=icecream

yes
[?- eats(tim, fries).
no
?-
```

4) Who are cousins?

**parent(min,gre).
parent(per,gre).
parent(tag,bound).
parent(upper,bound).
siblings(min,upper).
cousin(X,Y) :- parent(U,X), parent(V,Y), siblings(U,V).**

- (1) 1 Call: cousin(_0, _1)?
- (1) 1 Call: parent(_2, _0)?
- (1) 1 Exit: parent(min, gre)
- (2) 1 Call: parent(_2, _1)?
- (2) 1 Exit: parent(min, gre)
- (3) 1 Call: siblings(min, min)?
- (3) 1 Fail: siblings(min, min)
- (2) 1 Redo: parent(_2, _1)?
- (2) 1 Exit: parent(per, gre)
- (4) 1 Call: siblings(min, per)?
- (4) 1 Fail: siblings(min, per)
- (3) 1 Redo: parent(_2, _1)?
- (3) 1 Exit: parent(tag, bound)
- (5) 1 Call: siblings(min, tag)?
- (5) 1 Fail: siblings(min, tag)
- (4) 1 Redo: parent(_2, _1)?
- (4) 1 Exit: parent(upper, bound)
- (6) 1 Call: siblings(min, upper)?
- (6) 1 Exit: siblings(min, upper)
- (1) 1 Exit: cousin(gre, bound)