CS113 LAB 2: Truth Values Generator

SML

```
PolyML.print_depth 64;
exception OutOfRange;
fun expand((a:'a,b:'a),[]) = [] |
    expand((a:'a,b:'a), q::qs : 'a list list) =
    [q @ [a]] @ [q @ [b]] @ expand ((a,b), qs) : 'a list list;
fun tv(0, (_, _)) = [[]] : 'a list list |
    tv(1, (a:'a,b:'a)) = [[a],[b]] |
    tv(d, (a:'a,b:'a)) = expand((a,b), tv(d-1, (a,b)));
```

```
> fun expand((a:'a,b:'a),[]) = [] |
    expand((a:'a,b:'a), q::qs : 'a list list) =
    [q @ [a]] @ [q @ [b]] @ expand ((a,b), qs) : 'a list list;
fun tv(0, (_, _)) = [[]] : 'a list list |
    tv(1, (a:'a,b:'a)) = [[a],[b]] |
    tv(d, (a:'a,b:'a)) = expand((a,b), tv(d-1, (a,b)));
# # val expand = fn: ('a * 'a) * 'a list list -> 'a list list
> # # val tv = fn: int * ('a * 'a) -> 'a list list
```

CS113 LAB 2: NOT gate

Normal

P	$\sim P$
0	1
1	0

```
> map NOT (tv(1,(0,1)));
val it = [1, 0]: int list
```

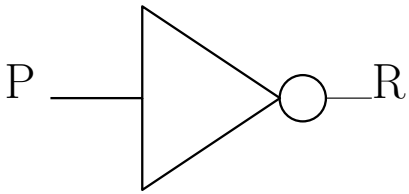
Reverse

P	$\sim P$
1	0
0	1

```
> map NOT (tv(1,(1,0)));
val it = [0, 1]: int list
```

SML

```
fun NOT [0] = 1 |
    NOT [1] = 0 |
    NOT [_] = raise OutOfRange;
map NOT (tv(1,(0,1)));
map NOT (tv(1,(1,0)));
```



CS113 LAB 2: AND gate

Normal

P	Q	$(P \wedge Q)$
0	0	0
0	1	0
1	0	0
1	1	1

```
> map pAND (tv(2,(0,1)));
val it = [0, 0, 0, 1]: int list
```

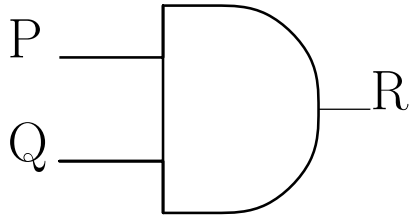
Reverse

P	Q	$(P \wedge Q)$
1	1	1
1	0	0
0	1	0
0	0	0

```
> map pAND (tv(2,(1,0)));
val it = [1, 0, 0, 0]: int list
```

SML

```
infix AND;
fun 0 AND _ = 0 |
    1 AND Q = Q |
    _ AND _ = raise OutOfRange;
fun pAND[P,Q] = P AND Q;
map pAND (tv(2,(0,1)));
map pAND (tv(2,(1,0)));
```



CS113 LAB 2: OR gate

Normal

P	Q	$(P \vee Q)$
0	0	0
0	1	1
1	0	1
1	1	1

```
> map pOR (tv(2,(0,1)));
val it = [0, 1, 1, 1]: int list
```

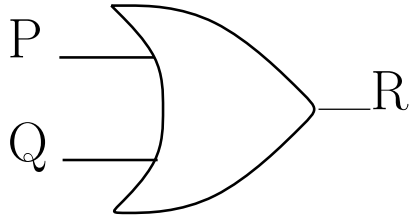
Reverse

P	Q	$(P \vee Q)$
1	1	1
1	0	1
0	1	1
0	0	0

```
> map pOR (tv(2,(1,0)));
val it = [1, 1, 1, 0]: int list
```

SML

```
infix OR;
fun 1 OR _ = 1 |
    0 OR Q = Q |
    _ OR _ = raise OutOfRange;
fun pOR[P,Q] = P OR Q;
map pOR (tv(2,(0,1)));
map pOR (tv(2,(1,0)));
```



CS113 LAB 2: NAND gate

Normal

P	Q	$\sim(P \wedge Q)$
0	0	1
0	1	1
1	0	1
1	1	0

```
> map pNAND (tv(2,(0,1)));
val it = [1, 1, 1, 0]: int list
```

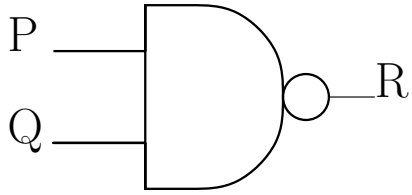
Reverse

P	Q	$\sim(P \wedge Q)$
1	1	0
1	0	1
0	1	1
0	0	1

```
> map pNAND (tv(2,(1,0)));
val it = [0, 1, 1, 1]: int list
```

SML

```
infix NAND;
fun P NAND Q = NOT [P AND Q];
fun pNAND[P,Q] = P NAND Q;
map pNAND (tv(2,(0,1)));
map pNAND (tv(2,(1,0)));
```



CS113 LAB 2: NOR gate

Normal

P	Q	$\sim(P \vee Q)$
0	0	1
0	1	0
1	0	0
1	1	0

```
> map pNOR (tv(2,(0,1)));
val it = [1, 0, 0, 0]: int list
```

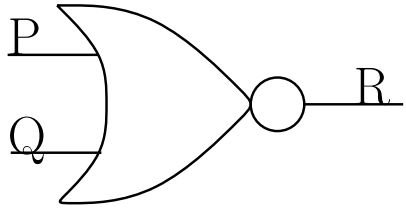
Reverse

P	Q	$\sim(P \vee Q)$
1	1	0
1	0	0
0	1	0
0	0	1

```
> map pNOR (tv(2,(1,0)));
val it = [0, 0, 0, 1]: int list
```

SML

```
infix NOR;
fun P NOR Q = NOT [P OR Q];
fun pNOR[P,Q] = P NOR Q;
map pNOR (tv(2,(0,1)));
map pNOR (tv(2,(1,0)));
```



CS113 LAB 2: XOR gate

Normal

P	Q	$(P \oplus Q)$
0	0	0
0	1	1
1	0	1
1	1	0

```
> map pXOR (tv(2,(0,1)));
val it = [0, 1, 1, 0]: int list
```

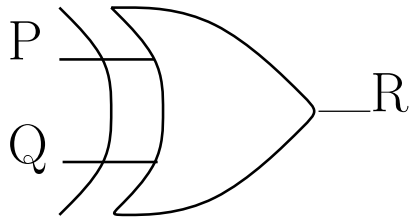
Reverse

P	Q	$(P \oplus Q)$
1	1	0
1	0	1
0	1	1
0	0	0

```
> map pXOR (tv(2,(1,0)));
val it = [0, 1, 1, 0]: int list
```

SML

```
infix XOR;
fun 0 XOR Q = Q |
    Q XOR 0 = Q |
    1 XOR 1 = 0 |
    _ XOR _ = raise OutOfRange;
fun pXOR[P,Q] = P XOR Q;
map pXOR (tv(2,(0,1)));
map pXOR (tv(2,(1,0)));
```



CS113 LAB 2: Example 2.2 & 2.3

Normal

P	Q	R	$(P \vee Q)$	$(P \vee R)$	$((P \vee Q) \wedge (P \vee R))$
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

```
> map ex2 (tv(3,(0,1)));
val it = [0, 0, 0, 1, 1, 1, 1, 1]: int list
```

Reverse

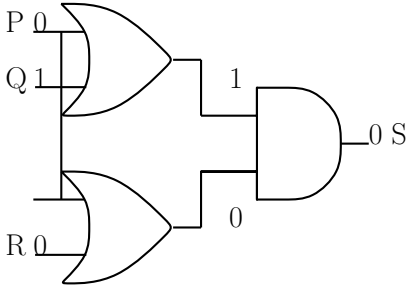
P	Q	R	$(P \vee Q)$	$(P \vee R)$	$((P \vee Q) \wedge (P \vee R))$
1	1	1	1	1	1
1	1	0	1	1	1
1	0	1	1	1	1
1	0	0	1	1	1
0	1	1	1	1	1
0	1	0	1	0	0
0	0	1	0	1	0
0	0	0	0	0	0

```
> map ex2 (tv(3,(1,0)));
val it = [1, 1, 1, 1, 1, 0, 0, 0]: int list
```

SML

```
fun ex2[P,Q,R] = (P OR Q) AND (P OR R);
map ex2 (tv(3,(0,1)));
map ex2 (tv(3,(1,0)));
```

P = 0, Q = 1, R = 0



CS113 LAB 2: Problem 2

Normal

P	Q	R	$(P \wedge Q)$	$\sim R$	$((P \wedge Q) \vee \sim R)$
0	0	0	0	1	1
0	0	1	0	0	0
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	1	0	1

```
> map p2 (tv(3,(0,1)));
val it = [1, 0, 1, 0, 1, 0, 1, 1]: int list
```

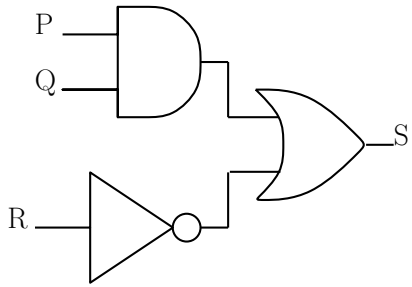
Reverse

P	Q	R	$(P \wedge Q)$	$\sim R$	$((P \wedge Q) \vee \sim R)$
1	1	1	1	0	1
1	1	0	1	1	1
1	0	1	0	0	0
1	0	0	0	1	1
0	1	1	0	0	0
0	1	0	0	1	1
0	0	1	0	0	0
0	0	0	0	1	1

```
> map p2 (tv(3,(1,0)));
val it = [1, 1, 0, 1, 0, 1, 0, 1]: int list
```

SML

```
fun p2[P,Q,R] = (P AND Q) OR NOT [R];
map p2 (tv(3,(0,1)));
map p2 (tv(3,(1,0)));
```



CS113 LAB 2: Problem 3

Normal

P	Q	R	$(P \wedge Q)$	$\sim R$	$((P \wedge Q) \wedge \sim R)$
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	0	1	0
0	1	1	0	0	0
1	0	0	0	1	0
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	1	0	0

```
> map p3 (tv(3,(0,1)));
val it = [0, 0, 0, 0, 0, 0, 1, 0]: int list
```

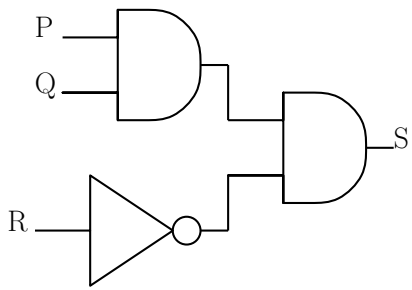
Reverse

P	Q	R	$(P \wedge Q)$	$\sim R$	$((P \wedge Q) \wedge \sim R)$
1	1	1	1	0	0
1	1	0	1	1	1
1	0	1	0	0	0
1	0	0	0	1	0
0	1	1	0	0	0
0	1	0	0	1	0
0	0	1	0	0	0
0	0	0	0	1	0

```
> map p3 (tv(3,(1,0)));
val it = [0, 1, 0, 0, 0, 0, 0, 0]: int list
```

SML

```
fun p3[P,Q,R] = (P AND Q) AND NOT [R];
map p3 (tv(3,(0,1)));
map p3 (tv(3,(1,0)));
```



CS113 LAB 2: Problem 7

Normal

A	B	S	$\sim S$	$(A \wedge \sim S)$	$(S \wedge B)$	$((A \wedge \sim S) \vee (S \wedge B))$
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	0	0	0
1	1	0	1	1	0	1
1	1	1	0	0	1	1

```
> map p7 (tv(3,(0,1)));
val it = [0, 0, 0, 1, 1, 0, 1, 1]: int list
```

Reverse

A	B	S	$\sim S$	$(A \wedge \sim S)$	$(S \wedge B)$	$((A \wedge \sim S) \vee (S \wedge B))$
1	1	1	0	0	1	1
1	1	0	1	1	0	1
1	0	1	0	0	0	0
1	0	0	1	1	0	1
0	1	1	0	0	1	1
0	1	0	1	0	0	0
0	0	1	0	0	0	0
0	0	0	1	0	0	0

```
> map p7 (tv(3,(1,0)));
val it = [1, 1, 0, 1, 1, 0, 0, 0]: int list
```

SML

```
fun p7[A,B,S] = (A AND NOT [S]) OR (S AND B);
map p7 (tv(3,(0,1)));
map p7 (tv(3,(1,0)));
```