```
Standard ML of New Jersey v110.78 [built: Thu Jul 23 11:21:58 2015]

- exec();
[opening project4.sml]
val it = () : unit
val it = () : unit
infix -->
infix v
infix &
infix <->
datatype sentence
  = & of sentence * sentence
  | --> of sentence * sentence
  | <-> of sentence * sentence
  | P
  | Q
  | R
  | S
  | T
  | v of sentence * sentence
  | ~ of sentence
val removeArrows = fn : sentence -> sentence
val bringInNegation = fn : sentence -> sentence
val distributeDisjInConj = fn : sentence -> sentence
val cnf = fn : sentence -> sentence
val cnf_1 = fn : sentence -> sentence
val show2 = fn : sentence -> unit
val show = fn : sentence -> unit
val run = fn : sentence -> unit
val printNStr = fn : string * int -> unit
val go1 = fn : int * int * sentence list -> unit
val go = fn : sentence list -> unit
val getConjuncts = fn : sentence -> unit
val listCNFs = fn : sentence list -> sentence list
val verifyCNFs = fn : sentence list * int -> unit
val exec = fn : unit -> unit
val f1 = P : sentence
val f2 = ~ P : sentence
val f3 = ~ (~ P) : sentence
val f4 = ~ (~ (~ P)) : sentence
val f5 = P v ~ P : sentence
val f6 = P --> Q : sentence
val f7 = P <-> Q : sentence
val f8 = P v Q --> P : sentence
val f9 = S & T v (Q & R) : sentence
val f10 = ~ S & ~ T : sentence
val f11 = ~ (P --> (~ Q --> ~ P)) : sentence
val f12 = P --> Q & (Q --> R) : sentence
val f13 = P --> Q & (Q --> R) --> (P --> R) : sentence
val f14 = ~ (P --> ~ Q v (~ P & ~ Q)) : sentence
val f15 = P & Q --> P : sentence
val f16 = P & Q v (R & S) : sentence
```

```
val f17 = P --> Q --> (~ Q --> ~ P) : sentence
val f18 = P --> ~ Q v (~ P & ~ Q) : sentence
val f19 = P --> Q <-> (~ Q --> ~ P) : sentence
val f20 = ~ (P --> ~ Q <-> (~ P & ~ Q)) : sentence
val f21 = ~ (P --> ~ Q v (~ P & ~ Q)) : sentence
val f22 = ~ (~ (P --> Q & (Q --> R) --> (P --> R))) : sentence
val f23 = P --> Q v (Q --> R) : sentence
val f24 = P --> Q & (Q --> R) --> (P --> R) : sentence
val f25 = P & Q v (~ P & ~ R) v (S & T v (~ Q & ~ P)) : sentence
val tests =
  [P,~ P,~ (~ P),~ (~ (~ P)),P v ~ P,P --> Q,P <-> Q,P v Q --> P,
   S & T v (Q & R),~ S & ~ T,~ (P --> (~ Q --> ~ P)),P --> Q & (Q --> R),
   P --> Q & (Q --> R) --> (P --> R),~ (P --> ~ Q v (~ P & ~ Q)),P & Q --> P,
   P & Q v (R & S),P --> Q --> (~ Q --> ~ P),P --> ~ Q v (~ P & ~ Q),
   P --> Q <-> (~ Q --> ~ P),~ (P --> ~ Q <-> (~ P & ~ Q)),
   ~ (P --> ~ Q v (~ P & ~ Q)),~ (~ (P --> Q & (Q --> R) --> (P --> R))),
   P --> Q v (Q --> R),P --> Q & (Q --> R) --> (P --> R),
   P & Q v (~ P & ~ R) v (S & T v (~ Q & ~ P))] : sentence list
val it = () : unit
- go tests;
```

==========

Formula #1

Sentence is: P

Its CNF is : P

==========

Formula #2

Sentence is: -P

Its CNF is : -P

==========

Formula #3

Sentence is: --P

Its CNF is : P

==========

Formula #4

Sentence is: ---P

Its CNF is : -P

==========

Formula #5

Sentence is: P v -P

Its CNF is : P v -P

==========

Formula #6

Sentence is: P -> Q

Its CNF is : -P v Q

==========

Formula #7

Sentence is: P <-> Q

Its CNF is : ((P v -P) & (Q v -P)) & ((P v -Q) & (Q v -Q))

==========

Formula #8

Sentence is: (P v Q) -> P

Its CNF is : (-P v P) & (-Q v P)

==========

Formula #9

Sentence is: (S & T) v (Q & R)

Its CNF is : ((S v Q) & (T v Q)) & ((S v R) & (T v R))

==========

Formula #10

Sentence is: -S & -T

Its CNF is : -S & -T

==========

Formula #11

Sentence is: -(P -> (-Q -> -P))

Its CNF is : P & (-Q & P)

==========

Formula #12

Sentence is: (P -> Q) & (Q -> R)

Its CNF is : (-P v Q) & (-Q v R)

==========

Formula #13

Sentence is: ((P -> Q) & (Q -> R)) -> (P -> R)

Its CNF is : (((P v Q) v (-P v R)) & ((-Q v Q) v (-P v R))) & (((P v -R) v (-P v R)) & ((-Q v -R) v (-P v R)))

==========

Formula #14

Sentence is: -((P -> -Q) v (-P & -Q))

Its CNF is : (P & Q) & (P v Q)

==========

Formula #15

Sentence is: (P & Q) -> P

Its CNF is : (-P v -Q) v P

==========

Formula #16

Sentence is: (P & Q) v (R & S)

Its CNF is : ((P v R) & (Q v R)) & ((P v S) & (Q v S))

==========

Formula #17

Sentence is: (P -> Q) -> (-Q -> -P)

Its CNF is : (P v (Q v -P)) & (-Q v (Q v -P))

==========

Formula #18

Sentence is: (P -> -Q) v (-P & -Q)

Its CNF is : ((-P v -Q) v -P) & ((-P v -Q) v -Q)

==========

Formula #19

Sentence is: (P -> Q) <-> (-Q -> -P)

Its CNF is : ((((-P v Q) v P) & ((Q v -P) v P)) & (((-P v Q) v -Q) & ((Q v -P) v -Q))) &
(((((-P v Q) v -Q) & ((Q v -P) v -Q)) & (((-P v Q) v P) & ((Q v -P) v P)))

==========

Formula #20

Sentence is: -((P -> -Q) <-> (-P & -Q))

Its CNF is : ((P v (P v Q)) & (Q v (P v Q))) & (((-P v -Q) v -P) & ((-P v -Q) v -Q))

==========

Formula #21

Sentence is: -((P -> -Q) v (-P & -Q))

Its CNF is : (P & Q) & (P v Q)

==========

Formula #22

Sentence is: -(-(((P -> Q) & (Q -> R)) -> (P -> R)))

Its CNF is : (((P v Q) v (-P v R)) & ((-Q v Q) v (-P v R))) & (((P v -R) v (-P v R)) & ((-Q v -R) v (-P v R)))

==========

Formula #23

Sentence is: (P -> Q) v (Q -> R)

Its CNF is : (-P v Q) v (-Q v R)

==========

Formula #24

Sentence is: ((P -> Q) & (Q -> R)) -> (P -> R)

Its CNF is : (((P v Q) v (-P v R)) & ((-Q v Q) v (-P v R))) & (((P v -R) v (-P v R)) & ((-Q v -R) v (-P v R)))

==========

Formula #25

Sentence is: ((P & Q) v (-P & -R)) v ((S & T) v (-Q & -P))

Its CNF is : (((((P v -P) v (S v -Q)) & ((Q v -P) v (S v -Q))) & (((P v -R) v (S v -Q)) & ((Q v -R) v (S v -Q)))) &
(((((P v -P) v (T v -Q)) & ((Q v -P) v (T v -Q))) & (((P v -R) v (T v -Q)) & ((Q v -R) v (T v -Q)))) &
(((((P v -P) v (S v -P)) & ((Q v -P) v (S v -P))) & (((P v -R) v (S v -P)) & ((Q v -R) v (S v -P)))) &
((((P v -P) v (T v -P)) & ((Q v -P) v (T v -P))) & (((P v -R) v (T v -P)) & ((Q v -R) v (T v -P)))))

==========

val it = () : unit

-