EX.NO:2	**Assign the truth table actions to decisions using suitable mathematical software**
DATE:9-8-2021	

20MA302 MATHEMATICAL STRUCTURES

OBJECTIVE:

To make decisions on the validity of logical statement by assigning the truth table for using mathematica.

SOFTWARE REQUIRED:

Execute Wolframcloud notebook (https://www.wolframcloud.com)

QUESTION 1:

Write a program to check whether the statements (p v q) and (q v p) are equivalent.

To set up function:

```
TruthTable[op_, n_] := Module[ { l = Flatten[Outer[List, Sequence @@
Table[{True, False}, {n}]], n - 1], a = Array[A, n] }, DisplayForm[
GridBox[Prepend[Append[#, op @@ #]& /@ l, Append[a, op @@ a]],
RowLines -> True, ColumnLines -> True] ] ]
A[1]=p;
A[2]=q;
```

CODING

```
TruthTable[Equivalent[Or[#1, #2], Or[#2, #1]] &, 2]

Print["Since all the Truth values are true, the given statements are
equivalent"]
```

OUTPUT

p	q	$p \| q \Leftrightarrow q \| p$
True	False	True

True	False	True
False	True	True
False	False	True

Since all the Truth values are true, the given statements are equivalent

QUESTION 2:

Generate the truth table to portray the validity of Complement Law.

<u>To set up function:</u>

```
TruthTable[op_, n_] := Module[ { l = Flatten[Outer[List, Sequence @@
Table[{True, False}, {n}]], n - 1], a = Array[A, n] }, DisplayForm[
GridBox[Prepend[Append[#, op @@ #]& /@ l, Append[a, op @@ a]],
RowLines -> True, ColumnLines -> True] ] ]
A[1]=p;
```

CODING

```
TruthTable[Equivalent[Or[#1, Not[#1]],True] &, 1]

TruthTable[Equivalent[#1&& Not[#1],False] &, 1]

Print["Since all the Truth values are true, Complement Law is valid"]
```

OUTPUT

p	$p \| ! p$
True	True
False	True

p	$!(p \&\& ! p)$
True	True
False	True

Since all the Truth values are true, Complement Law is valid.

QUESTION 3:

Generate the truth table to portray the validity of Distributive Law.

$P \vee (Q \wedge R) \Leftrightarrow (P \vee Q) \wedge (P \vee R)$.

<u>To set up function:</u>

```
TruthTable[op_, n_] := Module[ { l = Flatten[Outer[List, Sequence @@
Table[{True, False}, {n}]], n - 1], a = Array[A, n] }, DisplayForm[
GridBox[Prepend[Append[#, op @@ #]& /@ l, Append[a, op @@ a]],
RowLines -> True, ColumnLines -> True] ] ]
A[1]=p;
A[2]=q;
A[3]=r;
```

CODING

```
TruthTable[Equivalent[Or[#1,And[#2,#3]],And[Or[#1,#2],Or[#1,#3]]]&,
3]
```

```
Print["Since all the Truth values are true, Distributive Law is
valid"]
```

OUTPUT

p	q	r	$(p \| q) \&\& (p \| r) \Leftrightarrow p \| (q \&\& r)$
True	True	True	True
True	True	False	True
True	False	True	True
True	False	False	True
False	True	True	True
False	True	False	True
False	False	True	True
False	False	False	True

Since all the Truth values are true, Distributive Law is valid.

Conclusion :

Mathematica represents Boolean expressions in symbolic form, so they can not only be evaluated, but also be symbolically manipulated and transformed. Incorporating state-of-the-art quantifier elimination, satisfiability, and equational logic theorem proving, the Mathematica provides a powerful framework for investigations based on Boolean algebra.

<u>Problems for Practice</u>

20MA302 MATHEMATICAL STRUCTURES

LAB EXPERIMENT 1

1. Write a program to check whether the statements $p \wedge (p \Rightarrow q)$ and q are equivalent.

```
TruthTable[op_, n_] := Module[ { l = Flatten[Outer[List, Sequence @@
Table[{True, False}, {n}]], n - 1], a = Array[A, n] }, DisplayForm[
GridBox[Prepend[Append[#, op @@ #] &amp; /@ l, Append[a, op @@ a]],
RowLines → True, ColumnLines → True] ] ]
A[1] = p;
A[2] = q;
TruthTable[And[#1, Implies[#1, #2]] &, 2]
GridBox[{{p, q, p && (p ⇒ q)}, Null[{True, True}], Null[{True, False}],
  Null[{False, True}], Null[{False, False}]}, RowLines → True, ColumnLines → True]
```

2. Write a program to check whether the statements $p \wedge p$ and p are equivalent.

```
TruthTable[op_, n_] := Module[ { l = Flatten[Outer[List, Sequence @@
Table[{True, False}, {n}]], n - 1], a = Array[A, n] }, DisplayForm[
GridBox[Prepend[Append[#, op @@ #] &amp; /@ l, Append[a, op @@ a]],
RowLines → True, ColumnLines → True] ] ]
A[1] = p;
TruthTable[Equivalent[And[#1, #1], True] &, 1]

GridBox[{{p, p && p}, Null[{True}], Null[{False}]}, RowLines → True,
 ColumnLines → True]
```

3. Generate the truth table to portray the validity of Absorption Law.

```
TruthTable[op_, n_] := Module[{l = Flatten[Outer[List, Sequence @@
Table[{True, False}, {n}]], n - 1], a = Array[A, n]}, DisplayForm[
GridBox[Prepend[Append[#, op @@ #] &amp; /@ l, Append[a, op @@ a]],
RowLines → True, ColumnLines → True]]]
A[1] = p;
A[2] = q;
TruthTable[Equivalent[Or[#1, And[#1, #2]], #1] &, 2]

GridBox[{{p, q, p ⇔ p || (p && q)}, Null[{True, True}], Null[{True, False}],
  Null[{False, True}], Null[{False, False}]}, RowLines → True, ColumnLines → True]
```

4. Generate the truth table to portray the validity of Contra positive Law.

```
TruthTable[op_, n_] :=
 Module[{l = Flatten[Outer[List, Sequence @@ Table[{True, False}, {n}]], n - 1],
   a = Array[A, n]},
  DisplayForm[GridBox[Prepend[Append[#, op @@ #] & /@ l, Append[a, op @@ a]],
    RowLines → True, ColumnLines → True]]]
A[1] = p;
A[2] = q;
TruthTable[Equivalent[Implies[#1, #2], Implies[! #2, ! #1]] &, 2]
```

p	q	(p ⇒ q) ⇔ (! q ⇒ ! p)
True	True	True
True	False	True
False	True	True
False	False	True

5. Generate the truth table to portray the validity of Distributive Law
$P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R)$.

```
TruthTable[op_, n_] := Module[{l = Flatten[Outer[List, Sequence @@
Table[{True, False}, {n}]], n - 1], a = Array[A, n]}, DisplayForm[
GridBox[Prepend[Append[#, op @@ #] &amp; /@ l, Append[a, op @@ a]],
RowLines → True, ColumnLines → True]]]
A[1] = p;
A[2] = q;
A[2] = r;
TruthTable[Equivalent[And[#1, Or[#2, #3]], Or[And[#1, #2], And[#2 #3]]] &, 3]

GridBox[{{p, r, r, p && (r || r) ⇔ (p && r) || r^2}, Null[{True, True, True}],
  Null[{True, True, False}], Null[{True, False, True}], Null[{True, False, False}],
  Null[{False, True, True}], Null[{False, True, False}], Null[{False, False, True}],
  Null[{False, False, False}]}, RowLines → True, ColumnLines → True]
```

6. Write a program to check whether the statements $p \Rightarrow (q \Rightarrow r)$ and $(p \wedge q) \Rightarrow r$ are equivalent.

```
TruthTable[op_, n_] :=
 Module[{l = Flatten[Outer[List, Sequence @@ Table[{True, False}, {n}]], n - 1],
   a = Array[A, n]},
  DisplayForm[GridBox[Prepend[Append[#, op @@ #] & /@ l, Append[a, op @@ a]],
    RowLines → True, ColumnLines → True]]]
A[1] = p;
A[2] = q;
A[3] = r;
TruthTable[Equivalent[Implies[#1, Implies[#2, #3]], Implies[And[#1, #2], #3]] &,
 3]
```

p	q	r	(p ⇒ (q ⇒ r)) ⇔ (p && q ⇒ r)
True	True	True	True
True	True	False	True
True	False	True	True
True	False	False	True
False	True	True	True
False	True	False	True
False	False	True	True
False	False	False	True

7. Write a program to check whether the statements $p \Rightarrow (q \vee r)$ and $(p \Rightarrow q) \vee (p \Rightarrow r)$ are equivalent.

```
TruthTable[op_, n_] :=
 Module[{l = Flatten[Outer[List, Sequence @@ Table[{True, False}, {n}]], n - 1],
   a = Array[A, n]},
  DisplayForm[GridBox[Prepend[Append[#, op @@ #] & /@ l, Append[a, op @@ a]],
    RowLines → True, ColumnLines → True]]]
A[1] = p;
A[2] = q;
A[3] = r;
TruthTable[
 Equivalent[Implies[#1, Or[#2, #3]], Or[Implies[#1, #2], Implies[#1, #3]]] &, 3]
```

p	q	r	(p ⇒ q || r) ⇔ (p ⇒ q) || (p ⇒ r)
True	True	True	True
True	True	False	True
True	False	True	True
True	False	False	True
False	True	True	True
False	True	False	True
False	False	True	True
False	False	False	True

8. Using Call to the Truth table function, create the truth table for $(p \vee q) \Leftrightarrow (q \vee p)$

```
TruthTable[op_, n_] :=
 Module[{l = Flatten[Outer[List, Sequence @@ Table[{True, False}, {n}]], n - 1],
   a = Array[A, n]},
  DisplayForm[GridBox[Prepend[Append[#, op @@ #] & /@ l, Append[a, op @@ a]],
    RowLines → True, ColumnLines → True]]]
A[1] = p;
A[2] = q;
TruthTable[Equivalent[Or[#1, #2], Or[#2, #1]] &, 2]
```

p	q	p || q ⇔ q || p
True	True	True
True	False	True
False	True	True
False	False	True

9. Generate the truth table to portray the validity of Commutative Law using Conjunction.

```
TruthTable[op_, n_] :=
 Module[{l = Flatten[Outer[List, Sequence @@ Table[{True, False}, {n}]], n - 1],
   a = Array[A, n]},
  DisplayForm[GridBox[Prepend[Append[#, op @@ #] & /@ l, Append[a, op @@ a]],
    RowLines → True, ColumnLines → True]]]
A[1] = p;
A[2] = q;
TruthTable[Equivalent[And[#1, #2], And[#2, #1]] &, 2]
```

p	q	p && q ⇔ q && p
True	True	True
True	False	True
False	True	True
False	False	True

10. Generate the truth table to portray the validity of demorgons law.

```
TruthTable[op_, n_] :=
 Module[{l = Flatten[Outer[List, Sequence @@ Table[{True, False}, {n}]], n - 1],
   a = Array[A, n]},
  DisplayForm[GridBox[Prepend[Append[#, op @@ #] & /@ l, Append[a, op @@ a]],
    RowLines → True, ColumnLines → True]]]
A[1] = p;
A[2] = q;
TruthTable[Equivalent[Not[And[#1, #2]], Or[Not[#1], Not[#2]]] &, 2]
```

p	q	! (p && q) ⇔ ! p || ! q
True	True	True
True	False	True
False	True	True
False	False	True