SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

COIMBATORE – 641008.

(AN AUTONOMOUS INSTITUTION)
(ACCREDITED BY NAAC WITH ‘A’ GRADE | AFFILIATED TO ANNA UNIVERSITY)

DEPARTMENT OF SCIENCE AND HUMANITIES

21MA302 – MATHEMATICAL STRUCTURES LABORATORY RECORD BOOK

ACADEMIC YEAR 2022 - 2023

SUBMITTED BY

Name	**: RASIKA B**
Register No	**: 727721EUIT126**
Department / Section	**: INFORMATION TECHNOLOGY / C**
Year / Semester	**: II / III**

NOV/DEC 2022

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

COIMBATORE – 641008.

(AN AUTONOMOUS INSTITUTION)
(ACCREDITED BY NAAC WITH 'A' GRADE | AFFILIATED TO ANNA UNIVERSITY)

DEPARTMENT OF SCIENCE AND HUMANITIES

21MA302 – MATHEMATICAL STRUCTURES LABORATORY RECORD BOOK

PRACTICAL RECORD

Name	**: RASIKA B**
Register No	**: 727721EUIT126**
Department / Section	**: INFORMATION TECHNOLOGY / C**
Year / Semester	**: II / III**

BONAFIDE CERTIFICATE

Certified bonafide record work done by Mr./Ms.__________**_RASIKA B**_______________________,
Reg.No.**___727721EUIT126**__________________ during the academic year 2022-2023.

STAFF INCHARGE

Submitted for Anna university practical examination, held on _________________.

Index

S.No	Date	Name of the experiment	Marks	Staff Sign
1	18.8.22	Generate The Truth Table For Mathematical Logic Using Suitable Mathematical Software.		
2	18.8.22	Assign the truth table actions to decisions using suitable mathematical software.		
3	24.8.22	Examine the logical validity of arguments using suitable mathematical software.		
4	24.8.22	Testing the truth values of the statements by logical operators using MATLAB.		
5	26.8.22	Verification of De-Morgan’s Law using MATLAB.		
6	26.8.22	Set operations using MATLAB.		
7	26.8.22	Compute permutations functions using suitable mathematical software.		
8	26.8.22	Compute combinations functions using suitable mathematical software.		
9	5.9.22	Compute Prime numbers using MATLAB.		
10	5.9.22	Compute least common multiple of two integers using suitable mathematical software.		
11	5.9.22	Compute greatest common divisor of two integers using suitable mathematical software.		
12	5.9.22	Compute Quotient and remainder of two integers by division algorithm using suitable mathematical software.		

Faculty In-charge

Department of Science and Humanities

21MA302 MATHEMATICAL STRUCTURES

EX.NO:1	**Generate The Truth Table For Mathematical Logic Using Suitable Mathematical Software.**
DATE:18.8.22	

OBJECTIVE:

To generate the truth table for mathematical logic using mathematica.

SOFTWARE REQUIRED:

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

QUESTION 1:

Write a program to display all the possible output of Boolean expression p or q

CODING

```
BooleanTable[p || q, {p, q}]
```

OUTPUT

{True, True, True, False}

QUESTION 2:

Write a program to create the truth values of the Boolean expression $(x \vee y) \vee \sim x$

CODING

```
f = (x||y)||!x;
BooleanTable[{x,y }-> f, {x,y}]
```

OUTPUT

```
{{True,True}->True,{True,False}->True,{False,True}-
>True,{False,False}->True}
```

>>Tableform

```
True        True
True

True        True
False

False       True
True
```

```
False      True
False
```

QUESTION 3:

Using Call to the Truth table function, create the truth table for the operator CONJUCTION.

<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] ] ]
```

CODING

```
TruthTable[And[#1, #2] &, 2]
```

OUTPUT

A[1]	A[2]	A[1]&&A[2]
True	False	False
True	False	False
False	True	False
False	False	False

QUESTION 4:

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

<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;
```

CODING

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

OUTPUT

p	q	$(q \Rightarrow p) \Rightarrow (p \vee q)$
True	False	True
True	False	True
False	True	True
False	False	False

Problems for Practice

1. Write a program to display all the possible output of Boolean expression NOT P

```
BooleanTable[Not p]

{True, False}
```

2. Write a program to display all the possible output of Boolean expression $P \wedge (P \vee Q)$

```
f = (P AND (P || Q) )
BooleanTable[{P, Q } → f, {P, Q}]

AND P (P || Q)

{{True, True} → AND True², {True, False} → AND True², {False, True} → AND False True, {False, False} → AND
   False²}
```

3. Write a program to create the truth values of the Boolean expression $(x \wedge y) \vee \sim z$

```
f = ((X AND Y) || !Z)
BooleanTable[{X, Y, Z } → f, {X, Y, Z}]

AND X Y || !Z

{{True, True, True} → AND True², {True, True, False} → True, {True, False, True} → AND False True, {True,
    False, False} → True, {False, True, True} → AND False True, {False, True, False} → True, {False, False,
    True} → AND False², {False, False, False} → True}
```

4. Write a program to create the truth values of the Boolean expression $\sim (Q \Rightarrow R) \wedge R \wedge (P \Rightarrow Q)$

```
In[42]:= f = ! (Q → R) AND R AND (P → Q)
         BooleanTable[{P, Q, R} → f, {P, Q, R}]

Out[42]= ! AND² R (P → Q) (Q → R)

Out[43]= {{True, True, True} → ! AND² True (True → True)², {True, True, False} → ! AND² False (True → False) (True →
            True), {True, False, True} → ! AND² True (False → True) (True → False), {True, False, False} → ! AND² False
           (False → False) (True → False), {False, True, True} → ! AND² True (False → True) (True → True), {False, True,
            False} → ! AND² False (False → True) (True → False), {False, False, True} → ! AND² True (False → False) (False
            → True), {False, False, False} → ! AND² False (False → False)²}
```

5. Write a program to create the truth values of the Boolean expression $p \Rightarrow (q \vee p)$

```
In[44]:= f = P → (Q || P)
         BooleanTable[{P, Q} → f, {P, Q}]

Out[44]= P → Q || P

Out[45]= {{True, True} → True → True, {True, False} → True → True, {False, True} → False → True, {False, False} → False
           → False}
```

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.

EX.NO:2	**Assign the truth table actions to decisions using suitable mathematical software**
DATE: 18.8.22	

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.

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;
```

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.

Problems for Practice

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

In[50]:= 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;
TruthTable[Equivalent[And[#1, Implies[#1, #2]], #2] &, 2]
Print["Since all the Truth values are true, the given statements are equivalent"]
```

Out[50]= $Failed

Out[53]//DisplayForm=

p	q	q ⟺ p && (p ⇒ q)
True	True	True
True	False	True
False	True	False
False	False	True

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

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

In[62]:= 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;
TruthTable[Equivalent[And[#1, #1], #1] &, 1]
Print["Since all the Truth values are true, the given statements are equivalent"]
```

Out[62]= $Failed

Out[65]//DisplayForm=

p	p ⟺ p && p
True	True
False	True

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

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

```
In[22]:= 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, And[#1, #2]], #1] &, 2]
Print["Since all the Truth values are true, Absorption is law valid"]
```

Out[25]//DisplayForm=

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

Since all the Truth values are true, Absorption is law valid

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

```
In[87]:= 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;
TruthTable[Equivalent[Implies[#1, #2], Implies[Not[#2], Not[#1]]] &, 2]
Since all the Truth values are true, contrapositive is valid.
```

Out[90]//DisplayForm=

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

```
In[55]:= 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] ] ]

In[56]:= A[1] = p;

In[57]:= A[2] = q;

In[58]:= A[3] = r;
         TruthTable[Equivalent[And[#1, Or[#2, #3]], Or[And[#1, #2], And[#1, #3]]] &, 3]

Out[59]//DisplayForm=
```

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

```
In[60]:= Print["Since all the Truth values are true, Distributive Law is valid"]
         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.

EX.NO:3	Examine the logical validity of arguments using suitable mathematical software
DATE:24.8.22	

OBJECTIVE:

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

SOFTWARE REQUIRED:

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

QUESTION 1:

Write a program to check the validity of modus phonens.

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[[#1 && (#1® #2)]®#2 &, 2]
```

OUTPUT

p	q	(p&&p®q)® q
True	True	(True®True)®True
True	False	(True®false)®False
False	True	False®True
False	False	False®False

QUESTION 2:

Generate the truth table to Check the validity of the arguments,"If he is hungry then he can eat. If he eats then he will be happy. He is hungry. Therefore he is happy."

p: He is hungry, q: He can eat, r: He is happy.

The premises are p□q, q□r, p.

Conclusion : r

Truth table for [pэ(p□q)э(q□r)®r

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;
A[3]=r;
```

CODING

```
TruthTable[(#1&&(#1®#2)&&(#2®#3))®#3 &, 3]
```

OUTPUT

```
   p      q      r     p&&(p®q)&&(q®r)®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
```

Problems for Practice

1. Write a program to check the validity of (←p ∍ (p (q))®q.

```
In[53]:= 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]]]
         TruthTable[Implies[And[Not[#1], Or[#1, #2]], #2] &, 2]

Out[54]//DisplayForm=
            p      q    !p&&(p||q)⇒q
          True   True       True
          True   False      True
          False  True       True
          False  False      True
```

2. Write a program to check the validity of Chain Rule.

```
In[1]:= 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]]]

In[2]:= A[1] = p;

In[3]:= A[2] = q;

In[4]:= A[3] = r;

In[5]:= TruthTable[Implies[And[Implies[#1, #2], Implies[#2, #3]], Implies[#1, #3]] &, 3]
        Print["Chain rule"]

Out[5]//DisplayForm=
          p      q      r     (p ⇒ q) && (q ⇒ r) ⇒ (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
```

3. Write a program to check the validity of Modus Tollens.

```
In[14]:= 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] ] ]
       TruthTable[Implies[And[Implies[#1, #2], Not[#2]], Not[#1]] &, 2]
       Print[ "Modus Tollen" ]
```

Out[15]//DisplayForm=

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

4. Write a program to check the validity of (p∍←q) ™ ←(p□q).

```
In[27]:= 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] ] ]
       TruthTable[Equivalent[And[#1, Not[#2]], Not[Implies[#1, #2]]] &, 2]
```

Out[28]//DisplayForm=

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

5. Write a program to show that S(R is tautologically implied by (P(Q)϶(P□R)((Q□S).

```
In[]:= 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;
       A[4] = s;

In[74]:= TruthTable[Implies[And[Or[#1, #2], Implies[#1, #3], Implies[#2, #4]], Or[#4, #3]] &, 4]

Out[74]//DisplayForm=
```

p	q	r	s	(p || q) && (p ⇒ r) && (q ⇒ s) ⇒ s || r
True	True	True	True	True
True	True	True	False	True
True	True	False	True	True
True	True	False	False	True
True	False	True	True	True
True	False	True	False	True
True	False	False	True	True
True	False	False	False	True
False	True	True	True	True
False	True	True	False	True
False	True	False	True	True
False	True	False	False	True
False	False	True	True	True
False	False	True	False	True
False	False	False	True	True
False	False	False	False	True

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

EX.NO:4	**Testing the truth values of the statements by logical operators using MATLAB**
DATE:24.8.22	

OBJECTIVE:

To test the truth values of the statements by logical operators using MATLAB

SOFTWARE REQUIRED:

Execute MATLAB/Octave Online (GNU Octave, v4.2.1)

QUESTION NO: 1

Check whether x is larger than 8.

Solution:

MATLAB Program

```
>> x = 5;
>> x > 8
```

OUTPUT

```
ans =
 logical
   0 (false)
```

QUESTION NO: 2

Check whether x is not equal to 3.

Solution:

MATLAB Program

```
>> x = 5;
>> x ~= 3
```

OUTPUT

```
ans =
 logical
   1  (true)
```

<u>Problems for Practice</u>

<u>1.</u>Check whether x lies between -3 and 4.

a)x= 0, b) x= 4, c) x= -2

```
>> x=0;
x<=4 && x>=-3

ans =

  logical

   1
```

```
>> x=4;
x<=4 && x>=-3

ans =

  logical

   1
```

```
>> x=-2;
x<=4 && x>=-3

ans =

  logical

   1
```

```
>> 727721euit126
```

2.Check whether y elements [8 20 9 2 19 7 10] are larger than or equal to x elements x = [15 6 9 4 11 7 14]?

```
>> Y=[ 8 20 9 2 19 7 10] ;
X=[15 6 9 4 11 7 14];
Y>=X

ans =

  1×7 logical array

   0   1   1   0   1   1   0
```

3. Check whether a) x elements are equal to y elements.
b. x elements are not equal to y elements.
Given x = [8 20 9 2 19 7 10]; y = [8 -20 8 2 17 7 12];

```
>> x = [8 20 9 2 19 7 10];
y = [ 8 -20 8 2 17 7 12];
x==y

ans =

  1×7 logical array

   1   0   0   1   0   1   0

>> x = [8 20 9 2 19 7 10];
y = [ 8 -20 8 2 17 7 12];
x~=y

ans =

  1×7 logical array

   0   1   1   0   1   0   1

>> 72772leuitl26
```

4. Check the truth values for the given matrix $\begin{bmatrix} 2 & 9 & 4 \\ -3 & 5 & 2 \\ 6 & 7 & -1 \end{bmatrix}$ if a) x< = 2, b) x> -2

```
>> x=[2 9 4;-3 5 2;6 7 -1];
x>-2

ans =

  3×3 logical array

   1   1   1
   0   1   1
   1   1   1
```

```
>> x=[2 9 4;-3 5 2;6 7 -1];
x<=2

ans =

  3×3 logical array

   1   0   0
   1   0   1
   0   0   1

>> 72772leuit126
```

5. Check the truth values of the following," x is equal to 12 or x is less than 3" if a) x = 12, b) x = 11.

```
>> x=12;
x==12 || x<3

ans =

  logical

   1

>> x=11;
x==12 || x<3

ans =

  logical

   0
>> 72772leuit126
```

Conclusion :

MATLAB is an extraordinarily useful tool for all kinds of engineering analysis and design. Discrete mathematics in MATLAB are understandable computer programs and very easy to check the logical values.

EX.NO:5	**Verification of De-Morgan's Law using MATLAB**
DATE:26.8.22	

OBJECTIVE:

To verify De-Morgan's Law using MATLAB.

SOFTWARE REQUIRED:

Execute MATLAB/Octave Online (GNU Octave, v4.2.1)

QUESTION:

Verify De-Morgan's Law for the sets u=[1 2 3 4 5 6 7 8], a=[2 3 4 5] and b=[6 3 7 8]

Solution :

De-Morgan's Law:

$(A \cup B)' = A' \cap B'$ **(FIRST LAW)**

$(A \cap B)' = A' \cup B'$ **(SECOND LAW)**

MATLAB Program

>>u = [1 2 3 4 5 6 7 8]

>>a = [2 3 4 5]

>>b = [6 3 7 8]

>>c=union(a,b)

>> d=setdiff(u,c)   LHS OF FIRST LAW

>>e=setdiff(u,a)

>>f=setdiff(u,b)

>> g=intersect(e,f)   RHS OF FIRST LAW

>>h=intersect(a,b)

>> i=setdiff(u,a) LHS OF SECOND LAW

>>j=union(e,f)   RHS OF SECOND LAW

OUTPUT

u = 1 2 3 4 5 6 7 8

a = 2 3 4 5

b = 6 3 7 8

c = 2 3 4 5 6 7 8

d=1

e = 1 6 7 8

f = 1 2 4 5

g=1 // first law verified

h=3

i=1 2 4 5 6 7 8

j=1 2 4 5 6 7 8 //second law verified

Problems for Practice

1. Verify De-Morgan's Law for the sets u=[2 3 4 5 6 7 8 9 10], a=[7 8 9] and b=[6 7 8 10]

```
octave:13> u = [2 3 4 5 6 7 8 9 10]
a = [7 8 9]
b = [6 7 8 10]
c=union(a,b)
d=setdiff(u,c)
e=setdiff(u,a)
f=setdiff(u,b)
g=intersect(e,f)
h=intersect(a,b)
i=setdiff(u,h)
j=union(e,f)

u =

    2    3    4    5    6    7    8    9   10

a =

   7   8   9

b =

    6    7    8   10

c =

    6    7    8    9   10

d =

   2   3   4   5

e =

    2    3    4    5    6   10

f =

   2   3   4   5   9

g =

   2   3   4   5

h =

   7   8

i =

    2    3    4    5    6    9   10

j =

    2    3    4    5    6    9   10

octave:24> 727721euit126
```

2. Verify De-Morgan's Law for the sets u=[11 12 13 14 15 16 17 18], a=[12 13 14 15] and b=[16 13 17 18]

```
octave:1> u = [11 12 13 14 15 16 17 18]
a = [12 13 14 15]
b = [16 13 17 18]
c=union(a,b)
d=setdiff(u,c)
e=setdiff(u,a)
f=setdiff(u,b)
g=intersect(e,f)
h=intersect(a,b)
i=setdiff(u,h)
j=union(e,f)


u =

   11   12   13   14   15   16   17   18

a =

   12   13   14   15

b =

   16   13   17   18

c =

   12   13   14   15   16   17   18

d = 11
e =

   11   16   17   18

f =

   11   12   14   15

g = 11
h = 13
i =

   11   12   14   15   16   17   18

j =

   11   12   14   15   16   17   18

octave:12> 727721euit126
```

3. Verify De-Morgan's Law for the sets u=[1 2 3 4 5], a=[2 3 4] and b=[1 4 5]

```
octave:1> u = [1 2 3 4 5]
a = [2 3 4 ]
b = [1 4 5]
c=union(a,b)
d=setdiff(u,c)
e=setdiff(u,a)
f=setdiff(u,b)
g=intersect(e,f)
h=intersect(a,b)
i=setdiff(u,h)
j=union(e,f)

u =

   1   2   3   4   5

a =

   2   3   4

b =

   1   4   5

c =

   1   2   3   4   5

d = [](1x0)
e =

   1   5

f =

   2   3

g = [](1x0)
h = 4
i =

   1   2   3   5

j =

   1   2   3   5

octave:12> 727721euit126
```

4. Verify De-Morgan's Law for the sets u=[2 3 4 5 6 7 8 9], a=[2 3 4 5] and b=[6 3 7 8]

```
octave:1> u = [2 3 4 5 6 7 8 9]
a = [2 3 4 5 ]
b = [6 3 7 8]
c=union(a,b)
d=setdiff(u,c)
e=setdiff(u,a)
f=setdiff(u,b)
g=intersect(e,f)
h=intersect(a,b)
i=setdiff(u,h)
j=union(e,f)

u =

   2   3   4   5   6   7   8   9

a =

   2   3   4   5

b =

   6   3   7   8

c =

   2   3   4   5   6   7   8

d = 9
e =

   6   7   8   9

f =

   2   4   5   9

g = 9
h = 3
i =

   2   4   5   6   7   8   9

j =

   2   4   5   6   7   8   9

octave:12> 727721euit126
```

5. Verify De-Morgan's Law for the sets u=[1 2 3 4 5 6 7 8], a=[1 2 3] and b=[3 4 5]

```
octave:24> u = [1 2 3 4 5 6 7 8]
a = [1 2 3]
b = [3 4 5]
c=union(a,b)
d=setdiff(u,c)
e=setdiff(u,a)
f=setdiff(u,b)
g=intersect(e,f)
h=intersect(a,b)
i=setdiff(u,h)
j=union(e,f)

u =

   1   2   3   4   5   6   7   8

a =

   1   2   3

b =

   3   4   5

c =

   1   2   3   4   5

d =

   6   7   8

e =

   4   5   6   7   8

f =

   1   2   6   7   8

g =

   6   7   8

h = 3
i =

   1   2   4   5   6   7   8

j =

   1   2   4   5   6   7   8

octave:35> 727721euit126
```

6. Verify De-Morgan's Law for the sets u=[9 10 11 12 13 14 15], a=[12 13 15] and b=[12 14 16]

```
octave:1> u = [9 10 11 12 13 14 15]
a = [12 13 15]
b = [12 14 16]
c=union(a,b)
d=setdiff(u,c)
e=setdiff(u,a)
f=setdiff(u,b)
g=intersect(e,f)
h=intersect(a,b)
i=setdiff(u,h)
j=union(e,f)

u =

    9   10   11   12   13   14   15

a =

   12   13   15

b =

   12   14   16

c =

   12   13   14   15   16

d =

    9   10   11

e =

    9   10   11   14

f =

    9   10   11   13   15

g =

    9   10   11

h = 12
i =

    9   10   11   13   14   15

j =

    9   10   11   13   14   15

octave:12> 727721euit126
```

7. Verify De-Morgan’s Law for the sets u=[1,2,3, 4, …, 20], a=[1,2,3,….15] and b=[10,11,12,…20]

```
octave:1> u = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
a = [1 2 3 4 5 6 7 8 9 10 11 12 13 15]
b = [10 11 12 13 14 15 16 17 18 19 20]
c=union(a,b)
d=setdiff(u,c)
e=setdiff(u,a)
f=setdiff(u,b)
g=intersect(e,f)
h=intersect(a,b)
i=setdiff(u,h)
j=union(e,f)

u =

 Columns 1 through 16:

    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16

 Columns 17 through 20:

   17   18   19   20

a =

    1    2    3    4    5    6    7    8    9   10   11   12   13   15

b =

   10   11   12   13   14   15   16   17   18   19   20

c =

 Columns 1 through 16:

    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16

 Columns 17 through 20:

   17   18   19   20

d = [](1x0)
e =

   14   16   17   18   19   20

f =

   1   2   3   4   5   6   7   8   9

g = [](1x0)
h =

   10   11   12   13   15

i =

    1    2    3    4    5    6    7    8    9   14   16   17   18   19   20

j =

    1    2    3    4    5    6    7    8    9   14   16   17   18   19   20

octave:12> 727721euit126
error: parse error:
```

8. Verify De-Morgan's Law for the sets u=[1 2 5 6 7 8 9 10], a=[1 2 7 8] and b=[5 7 9 10]

```
octave:1> u = [1 2 5 6 7 8 9 10]
a = [1 2 7 8]
b = [5 7 9 10]
c=union(a,b)
d=setdiff(u,c)
e=setdiff(u,a)
f=setdiff(u,b)
g=intersect(e,f)
h=intersect(a,b)
i=setdiff(u,h)
j=union(e,f)
u =

    1    2    5    6    7    8    9   10

a =

   1   2   7   8

b =

    5    7    9   10

c =

    1    2    5    7    8    9   10

d = 6
e =

    5    6    9   10

f =

   1   2   6   8

g = 6
h = 7
i =

    1    2    5    6    8    9   10

j =

    1    2    5    6    8    9   10

octave:12> 727721euit126
```

9. Verify De-Morgan’s Law for the sets u=[1 2 3 4 5 6 7 8], a=[3 4 5] and b=[6 7 8]

```
octave:1> u = [1 2 3 4 5 6 7 8]
a = [3 4 5]
b = [6 7 8]
c=union(a,b)
d=setdiff(u,c)
e=setdiff(u,a)
f=setdiff(u,b)
g=intersect(e,f)
h=intersect(a,b)
i=setdiff(u,h)
j=union(e,f)
u =

   1   2   3   4   5   6   7   8

a =

   3   4   5

b =

   6   7   8

c =

   3   4   5   6   7   8

d =

   1   2

e =

   1   2   6   7   8

f =

   1   2   3   4   5

g =

   1   2

h = [](1x0)
i =

   1   2   3   4   5   6   7   8

j =

   1   2   3   4   5   6   7   8

octave:12> 727721euit126
```

10. Verify De-Morgan's Law for the sets u=[1 2 5 7 8 9 10], a=[1 2 8] and b=[5 7 9]

```
octave:1> u = [1 2 5 7 8 9 10]
a = [1 2 8]
b = [5 7 9]
c=union(a,b)
d=setdiff(u,c)
e=setdiff(u,a)
f=setdiff(u,b)
g=intersect(e,f)
h=intersect(a,b)
i=setdiff(u,h)
j=union(e,f)

u =

    1    2    5    7    8    9   10

a =

   1   2   8

b =

   5   7   9

c =

   1   2   5   7   8   9

d = 10
e =

    5    7    9   10

f =

    1    2    8   10

g = 10
h = [](1x0)
i =

    1    2    5    7    8    9   10

j =

    1    2    5    7    8    9   10

octave:12> 727721euit126
```

Conclusion :

MATLAB is an extraordinarily useful tool for all kinds of engineering analysis and design. Solving De-Morgan's Law MATLAB are understandable computer programs and very easy to evaluate. MATLAB has several indexing styles that are not only powerful and flexible, but also readable and expressive.

EX.NO:6	Set operations using MATLAB
DATE: 26.8.22	

OBJECTIVE:

To execute set operations using MATLAB.

SOFTWARE REQUIRED:

Execute MATLAB/Octave Online (GNU Octave, v4.2.1)

QUESTION:

Find the union, intersection, difference, complement of the sets u=[1 2 3 4 5 6 7 8], a=[2 3 4 5] and b=[6 3 7 8]

MATLAB Program

```
>> u = [1 2 3 4 5 6 7 8 ]

>> a = [2 3 4 5]

>>b = [6 7 8 9]

>>c=union(a,b)           //aUb

>>d=intersect(a,b)       //a∩b

>>e=setdiff(a,b)         //a-b

>>f=setdiff(b,a)         //b-a

>>g=setdiff(u,a)         //ā (complement of a)

>>h=setdiff(u,b)         //b̄ (complement of b)

>>i=setxor(a,b)          //aΔb (symmetric difference) (A-B)U(B-A)
```

OUTPUT

u =1 2 3 4 5 6 7 8

a = 2 3 4 5

b = 6 3 7 8

c=2 3 4 5 6 7 8

d=3

e = 2 4 5

f = 6 7 8

g = 1 6 7 8

h = 1 2 4 5

i= 2 4 5 6 7 8

Problems for Practice

1. Find the union, intersection, difference, complement of the sets u=[2 3 4 5 6 7 8 9 10], a=[7 8 9] and b=[6 7 8 10]

```
octave:1> u = [2 3 4 5 6 7 8 9 10]
a = [7 8 9]
b = [6 7 8 10]
c=union(a,b)
d=intersect(a,b)
e=setdiff(a,b)
f=setdiff(b,a)
g=setdiff(u,a)
h=setdiff(u,b)
i=setxor(a,b)
u =

    2    3    4    5    6    7    8    9   10

a =

   7   8   9

b =

    6    7    8   10

c =

    6    7    8    9   10

d =

   7   8

e = 9
f =

    6   10

g =

    2    3    4    5    6   10

h =

   2   3   4   5   9

i =

    6    9   10

octave:11> 727721euit126
```

2. Find the union, intersection, difference, complement of the sets u=[11 12 13 14 15 16 17 18], a=[12 13 14 15] and b=[16 13 17 18]

```
octave:11> u = [11 12 13 14 15 16 17 18]
a = [12 13 14 15]
b = [16 13 17 18]
c=union(a,b)
d=intersect(a,b)
e=setdiff(a,b)
f=setdiff(b,a)
g=setdiff(u,a)
h=setdiff(u,b)
i=setxor(a,b)
u =

   11   12   13   14   15   16   17   18

a =

   12   13   14   15

b =

   16   13   17   18

c =

   12   13   14   15   16   17   18

d = 13
e =

   12   14   15

f =

   16   17   18

g =

   11   16   17   18

h =

   11   12   14   15

i =

   12   14   15   16   17   18

octave:21> 727721euit126
```

3. Find the union, intersection, difference, complement of the sets u=[1 2 3 4 5], a=[2 3 4] and b=[1 4 5]

```
octave:21> u = [1 2 3 4 5]
a = [2 3 4]
b = [1 4 5]
c=union(a,b)
d=intersect(a,b)
e=setdiff(a,b)
f=setdiff(b,a)
g=setdiff(u,a)
h=setdiff(u,b)
i=setxor(a,b)

u =

   1   2   3   4   5

a =

   2   3   4

b =

   1   4   5

c =

   1   2   3   4   5

d = 4
e =

   2   3

f =

   1   5

g =

   1   5

h =

   2   3

i =

   1   2   3   5

octave:31> 727721euit126
```

4. Find the union, intersection, difference, complement of the sets u=[2 3 4 5 6 7 8 9], a=[2 3 4 5] and b=[6 3 7 8]

```
octave:31> u = [2 3 4 5 6 7 8 9]
a = [2 3 4 5]
b = [6 3 7 8]
c=union(a,b)
d=intersect(a,b)
e=setdiff(a,b)
f=setdiff(b,a)
g=setdiff(u,a)
h=setdiff(u,b)
i=setxor(a,b)
u =

   2   3   4   5   6   7   8   9

a =

   2   3   4   5

b =

   6   3   7   8

c =

   2   3   4   5   6   7   8

d = 3
e =

   2   4   5

f =

   6   7   8

g =

   6   7   8   9

h =

   2   4   5   9

i =

   2   4   5   6   7   8

octave:41> 727721euit126
```

5. Find the union, intersection, difference, complement of the sets u=[1 2 3 4 5 6 7 8], a=[1 2 3] and b=[3 4 5]

```
octave:41> u = [1 2 3 4 5 6 7 8]
a = [1 2 3]
b = [3 4 5]
c=union(a,b)
d=intersect(a,b)
e=setdiff(a,b)
f=setdiff(b,a)
g=setdiff(u,a)
h=setdiff(u,b)
i=setxor(a,b)
u =

   1   2   3   4   5   6   7   8

a =

   1   2   3

b =

   3   4   5

c =

   1   2   3   4   5

d = 3
e =

   1   2

f =

   4   5

g =

   4   5   6   7   8

h =

   1   2   6   7   8

i =

   1   2   4   5

octave:51> 727721euit126
```

6. Find the union, intersection, difference, complement of the sets u=[9 10 11 12 13 14 15], a=[12 13 15] and b=[12 14 16]

```
octave:51> u = [9 10 11 12 13 14 15]
a = [12 13 15]
b = [12 14 16]
c=union(a,b)
d=intersect(a,b)
e=setdiff(a,b)
f=setdiff(b,a)
g=setdiff(u,a)
h=setdiff(u,b)
i=setxor(a,b)
u =

    9   10   11   12   13   14   15

a =

   12   13   15

b =

   12   14   16

c =

   12   13   14   15   16

d = 12
e =

   13   15

f =

   14   16

g =

    9   10   11   14

h =

    9   10   11   13   15

i =

   13   14   15   16

octave:61> 727721euit126
```

7. Find the union, intersection, difference, complement of the sets u=[1,2,3, 4, …, 20], a=[1,2,3,….15] and b=[10,11,12,…20]

```
octave:61> u = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
a = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15]
b = [10 11 12 13 14 15 16 17 18 19 20]
c=union(a,b)
d=intersect(a,b)
e=setdiff(a,b)
f=setdiff(b,a)
g=setdiff(u,a)
h=setdiff(u,b)
i=setxor(a,b)

u =

 Columns 1 through 16:

    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16

 Columns 17 through 20:

   17   18   19   20

a =

    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15

b =

   10   11   12   13   14   15   16   17   18   19   20

c =

 Columns 1 through 16:

    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16

 Columns 17 through 20:

   17   18   19   20

d =

   10   11   12   13   14   15

e =

   1   2   3   4   5   6   7   8   9

f =

   16   17   18   19   20

g =

   16   17   18   19   20

h =

   1   2   3   4   5   6   7   8   9

i =

    1    2    3    4    5    6    7    8    9   16   17   18   19   20

octave:71> 727721euit126
```

8. Find the union, intersection, difference, complement of the sets u=[1 2 5 6 7 8 9 10], a=[1 2 7 8] and b=[5 7 9 10]

```
octave:71> u = [1 2 5 6 7 8 9 10]
a = [1 2 7 8]
b = [5 7 9 10]
c=union(a,b)
d=intersect(a,b)
e=setdiff(a,b)
f=setdiff(b,a)
g=setdiff(u,a)
h=setdiff(u,b)
i=setxor(a,b)
u =

    1    2    5    6    7    8    9   10

a =

   1   2   7   8

b =

    5    7    9   10

c =

    1    2    5    7    8    9   10

d = 7
e =

   1   2   8

f =

    5    9   10

g =

    5    6    9   10

h =

   1   2   6   8

i =

    1    2    5    8    9   10

octave:81> 727721euit126
```

9. Find the union, intersection, difference, complement of the sets u=[1 2 3 4 5 6 7 8], a=[3 4 5] and b=[6 7 8]

```
octave:81> u = [1 2 3 4 5 6 7 8]
a = [3 4 5]
b = [6 7 8]
c=union(a,b)
d=intersect(a,b)
e=setdiff(a,b)
f=setdiff(b,a)
g=setdiff(u,a)
h=setdiff(u,b)
i=setxor(a,b)

u =

   1   2   3   4   5   6   7   8

a =

   3   4   5

b =

   6   7   8

c =

   3   4   5   6   7   8

d = [](1x0)
e =

   3   4   5

f =

   6   7   8

g =

   1   2   6   7   8

h =

   1   2   3   4   5

i =

   3   4   5   6   7   8

octave:91> 727721euit126
```

10. Find the union, intersection, difference, complement of the sets u=[1 2 5 7 8 9 10], a=[1 2 8] and b=[5 7 9]

```
octave:91> u = [1 2 5 7 8 9 10]
a = [1 2 8]
b = [5 7 9]
c=union(a,b)
d=intersect(a,b)
e=setdiff(a,b)
f=setdiff(b,a)
g=setdiff(u,a)
h=setdiff(u,b)
i=setxor(a,b)
u =

    1    2    5    7    8    9   10

a =

   1   2   8

b =

   5   7   9

c =

   1   2   5   7   8   9

d = [](1x0)
e =

   1   2   8

f =

   5   7   9

g =

    5    7    9   10

h =

    1    2    8   10

i =

   1   2   5   7   8   9

octave:101> 727721euit126
```

Conclusion :

MATLAB is an extraordinarily useful tool for all kinds of engineering analysis and design. Solving set theory related concepts in MATLAB are understandable computer programs and very easy to evaluate. MATLAB has several indexing styles that are not only powerful and flexible, but also readable and expressive.

EX.NO:7	**Compute permutations functions using suitable mathematical software**
DATE: 26.8.22	

OBJECTIVE:

To compute permutations functions using MATLAB.

SOFTWARE REQUIRED:

Execute MATLAB/Octave Online (GNU Octave, v4.2.1)

SYNTEX:

All possible permutations **P= Perms(v)**

P = perms(v) returns a matrix containing all permutations of the elements of vector v in reverse lexicographic order. Each row of P contains a different permutation of the *n* elements in v. Matrix P has the same data type as v, and it has *n*! rows and *n* columns.

QUESTION NO: 1

Find all permutations of {1,2,3}

Solution:

MATLAB Program

```
v=[1 2 3]
perms(v)
size(perms(v),1)
```

OUTPUT

```
v=[1 2 3]
v =
   1   2   3
 perms(v)
 ans =

   3   2   1
   3   1   2
   2   3   1
   2   1   3
   1   3   2
   1   2   3
size(perms(v),1)
 ans = 6
```

QUESTION NO: 2   Find P(6,3)

Solution:

MATLAB Program

P(n,r)=n!/(n-r)!

```
a=factorial(n)/factorial(n-r)
```

OUTPUT

```
a=factorial(6)/factorial(3)
a = 120
```

Problems for Practice

1. Find all permutations of {112,2345,65743}

```
>> v=[112,2345,65743]
perms(v)
size(perms(v),1)

v =

     112        2345       65743


ans =

       65743        2345         112
       65743         112        2345
        2345       65743         112
        2345         112       65743
         112       65743        2345
         112        2345       65743


ans =

     6

>> 72772leuitl26
```

2. Find all permutations of {1+1i 2+1i 3+1i}

```
>> v=[1+1i 2+1i 3+1i]
perms(v)
size(perms(v),1)

v =

   1.0000 + 1.0000i   2.0000 + 1.0000i   3.0000 + 1.0000i


ans =

   3.0000 + 1.0000i   2.0000 + 1.0000i   1.0000 + 1.0000i
   3.0000 + 1.0000i   1.0000 + 1.0000i   2.0000 + 1.0000i
   2.0000 + 1.0000i   3.0000 + 1.0000i   1.0000 + 1.0000i
   2.0000 + 1.0000i   1.0000 + 1.0000i   3.0000 + 1.0000i
   1.0000 + 1.0000i   3.0000 + 1.0000i   2.0000 + 1.0000i
   1.0000 + 1.0000i   2.0000 + 1.0000i   3.0000 + 1.0000i


ans =

     6

>> 72772leuitl26
```

3. Find all permutations of int16([1 2 3 4])

```
>> v=[int16([1 2 3 4])]
perms(v)
size(perms(v),1)

v =

  1×4 int16 row vector

   1   2   3   4


ans =

  24×4 int16 matrix

   4   3   2   1
   4   3   1   2
   4   2   3   1
   4   2   1   3
   4   1   3   2
   4   1   2   3
   3   4   2   1
   3   4   1   2
   3   2   4   1
   3   2   1   4
   3   1   4   2
   3   1   2   4
   2   4   3   1
   2   4   1   3
   2   3   4   1
   2   3   1   4
   2   1   4   3
   2   1   3   4
   1   4   3   2
   1   4   2   3
   1   3   4   2
   1   3   2   4
   1   2   4   3
   1   2   3   4


ans =

    24

>> 72772leuitl26
```

4. Find all permutations of ['abcd']

```
>> v=[ 'abcd' ]
perms(v)
size(perms(v),1)

v =

    'abcd'


ans =

  24×4 char array

    'dcba'
    'dcab'
    'dbca'
    'dbac'
    'dacb'
    'dabc'
    'cdba'
    'cdab'
    'cbda'
    'cbad'
    'cadb'
    'cabd'
    'bdca'
    'bdac'
    'bcda'
   'bcad'
   'badc'
   'bacd'
   'adcb'
   'adbc'
   'acdb'
   'acbd'
   'abdc'
   'abcd'


ns =

    24
>> 72772leuitl26
```

5. Find P(8,1), P(12,3) & P(8,8)

```
>> a=factorial(8)/factorial(7)
a=factorial(12)/factorial(9)
a=factorial(8)/factorial(0)


a =

     8


a =

        1320


a =

       40320

>> 72772leuitl26
```

Conclusion :

MATLAB is an extraordinarily useful tool for all kinds of engineering analysis and design in MATLAB ideas in understandable computer programs and very easy to compute permutations functions.

EX.NO:8	Compute combinations functions using suitable mathematical software
DATE:26.8.22	

OBJECTIVE:

To compute combinations functions using MATLAB.

SOFTWARE REQUIRED:

Execute MATLAB/Octave Online (GNU Octave, v4.2.1)

SYNTEX:

All combinations **C= nchoosek(n,k)** C= nchoosek(v,k) returns a matrix containing all possible combinations of the elements of vector v taken k at a time. Matrix C has k columns and $m!/((m-k)!\,k!)$ rows, where m is length(v)

QUESTION NO: 1

All Combinations of Five Numbers Taken Four at a Time {2,4,6,8,10}

Solution:

MATLAB Program

```
v=[2 4 6 8 10]
C = nchoosek(v,4)
size(C,1)
```

OUTPUT

```
v=[2 4 6 8 10]
v =

    2    4    6    8   10
C = nchoosek(v,4)
C =

    2    4    6    8
    2    4    6   10
    2    4    8   10
    2    6    8   10
    4    6    8   10
size(C,1)
ans = 5
```

QUESTION NO: 2 Find C(5,4)

Solution:

MATLAB Program

```
b = nchoosek(5,4)
```

OUTPUT

```
b = nchoosek(5,4)
b = 5
```

Problems for Practice

1. Find all combinations of uint16([10 20 30])

```
octave:23>  v = uint16([10 20 30])
C = nchoosek(v,uint16(2))
size(C,1)
v =

  10  20  30

C =

  10  20
  10  30
  20  30

ans = 3
octave:26> 727721euit126
```

2. Find all combinations of {1+1i 2+1i 3+1i}

```
octave:4> v = [1+1i 2 +1i 3+1i]
C = nchoosek(v,4)
size(C,1)
v =

   1 + 1i   2 + 0i   0 + 1i   3 + 1i

C =

   1 + 1i   2 + 0i   0 + 1i   3 + 1i

ans = 1
octave:7> 727721euit126
```

3. Find all combinations of int16([1 2 3 4])

```
octave:4>  v = int16([1 2 3 4])
C = nchoosek(v,int16(2))
size(C,1)
v =

  1  2  3  4

C =

  1  2
  1  3
  1  4
  2  3
  2  4
  3  4

ans = 6
octave:7> 727721euit126
```

4. Find all combinations of ['abcd']

```
>> v = ['abcd']
C = nchoosek(v,4)
size(C,1)

v =

    'abcd'


C =

    'abcd'


ans =

     1

>> 727721euit126
```

5. Find C(8,1), C(5,3) & C(8,0)

```
octave:1> b = nchoosek(8,1)
b = 8
octave:2> 727721euit126
```

```
octave:3> b = nchoosek(5,3)
b = 10
octave:4> 727721euit126
```

```
octave:5> b = nchoosek(8,0)
b = 1
octave:6> 727721euit126
```

Conclusion :

MATLAB is an extraordinarily useful tool for all kinds of engineering analysis and design in MATLAB ideas in understandable computer programs and very easy to compute combination functions.

EX.NO:9	Compute Prime numbers using MATLAB .
DATE:5.9.22	

OBJECTIVE:

To Compute Prime numbers using MATLAB with examples.

SOFTWARE REQUIRED:

Execute MATLAB/Octave Online (GNU Octave, v4.2.1)

QUESTION NO: 1

Find prime numbers from 1 to 100.

MATLAB Program

INPUT

```
>>num = 1:100 ;

>>idx = isprime(num) ;

>>num(idx)
```

OUPUT:

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71
73 79 83 89 97
```

Problems for Practice

SOLVE THE FOLLOWING:

1. **Find prime numbers from 100 to 200.**

```
octave:1> num=100:200;
idx=isprime(num);
num(idx)
ans =

 Columns 1 through 13:

   101   103   107   109   113   127   131   137   139   149   151   157   163

 Columns 14 through 21:

   167   173   179   181   191   193   197   199

octave:4> 727721euit126
```

2. **Find prime numbers from 500 to 800.**

```
octave:1> num=500:800;
idx=isprime(num);
num(idx)
ans =

 Columns 1 through 13:

   503   509   521   523   541   547   557   563   569   571   577   587   593

 Columns 14 through 26:

   599   601   607   613   617   619   631   641   643   647   653   659   661

 Columns 27 through 39:

   673   677   683   691   701   709   719   727   733   739   743   751   757

 Columns 40 through 44:

   761   769   773   787   797

octave:4> 727721euit126
```

3. **Find prime numbers from 1000 to 2000.**

```
octave:1> num=1000:2000;
idx=isprime(num);
num(idx)
ans =

 Columns 1 through 11:

   1009   1013   1019   1021   1031   1033   1039   1049   1051   1061   1063

 Columns 12 through 22:

   1069   1087   1091   1093   1097   1103   1109   1117   1123   1129   1151

 Columns 23 through 33:

   1153   1163   1171   1181   1187   1193   1201   1213   1217   1223   1229

 Columns 34 through 44:

   1231   1237   1249   1259   1277   1279   1283   1289   1291   1297   1301

 Columns 45 through 55:

   1303   1307   1319   1321   1327   1361   1367   1373   1381   1399   1409

 Columns 56 through 66:

   1423   1427   1429   1433   1439   1447   1451   1453   1459   1471   1481

 Columns 67 through 77:

   1483   1487   1489   1493   1499   1511   1523   1531   1543   1549   1553

 Columns 78 through 88:

   1559   1567   1571   1579   1583   1597   1601   1607   1609   1613   1619

 Columns 89 through 99:

   1621   1627   1637   1657   1663   1667   1669   1693   1697   1699   1709

 Columns 100 through 110:

   1721   1723   1733   1741   1747   1753   1759   1777   1783   1787   1789

 Columns 111 through 121:

   1801   1811   1823   1831   1847   1861   1867   1871   1873   1877   1879

 Columns 122 through 132:

   1889   1901   1907   1913   1931   1933   1949   1951   1973   1979   1987

 Columns 133 through 135:

   1993   1997   1999

octave:4> 727721euit126
```

4. **Find prime numbers from 2500 to 3500.**

```
octave:1> num=2500:3500;
idx=isprime(num);
num(idx)
ans =

 Columns 1 through 11:

   2503   2521   2531   2539   2543   2549   2551   2557   2579   2591   2593

 Columns 12 through 22:

   2609   2617   2621   2633   2647   2657   2659   2663   2671   2677   2683

 Columns 23 through 33:

   2687   2689   2693   2699   2707   2711   2713   2719   2729   2731   2741

 Columns 34 through 44:

   2749   2753   2767   2777   2789   2791   2797   2801   2803   2819   2833

 Columns 45 through 55:

   2837   2843   2851   2857   2861   2879   2887   2897   2903   2909   2917

 Columns 56 through 66:

   2927   2939   2953   2957   2963   2969   2971   2999   3001   3011   3019

 Columns 67 through 77:

   3023   3037   3041   3049   3061   3067   3079   3083   3089   3109   3119

 Columns 78 through 88:

   3121   3137   3163   3167   3169   3181   3187   3191   3203   3209   3217

 Columns 89 through 99:

   3221   3229   3251   3253   3257   3259   3271   3299   3301   3307   3313

 Columns 100 through 110:

   3319   3323   3329   3331   3343   3347   3359   3361   3371   3373   3389

 Columns 111 through 121:

   3391   3407   3413   3433   3449   3457   3461   3463   3467   3469   3491

 Column 122:

   3499

octave:4> 727721euit126
```

5. **Find prime numbers from 9000 to 10000.**

```
octave:1> num=9000:10000;
idx=isprime(num);
num(idx)
ans =

 Columns 1 through 11:

   9001   9007   9011   9013   9029   9041   9043   9049   9059   9067   9091

 Columns 12 through 22:

   9103   9109   9127   9133   9137   9151   9157   9161   9173   9181   9187

 Columns 23 through 33:

   9199   9203   9209   9221   9227   9239   9241   9257   9277   9281   9283

 Columns 34 through 44:

   9293   9311   9319   9323   9337   9341   9343   9349   9371   9377   9391

 Columns 45 through 55:

   9397   9403   9413   9419   9421   9431   9433   9437   9439   9461   9463

 Columns 56 through 66:

   9467   9473   9479   9491   9497   9511   9521   9533   9539   9547   9551

 Columns 67 through 77:

   9587   9601   9613   9619   9623   9629   9631   9643   9649   9661   9677

 Columns 78 through 88:

   9679   9689   9697   9719   9721   9733   9739   9743   9749   9767   9769

 Columns 89 through 99:

   9781   9787   9791   9803   9811   9817   9829   9833   9839   9851   9857

 Columns 100 through 110:

   9859   9871   9883   9887   9901   9907   9923   9929   9931   9941   9949

 Columns 111 and 112:

   9967   9973

octave:4> 727721euit126
```

Conclusion :

We can say that this software and methodology offers strong possibilities, utilization is pretty easy for simple and advanced problems on finding prime numbers.

EX.NO:10	**Compute least common multiple of two integers using suitable mathematical software**
DATE:5.9.22	

OBJECTIVE:

To find least common multiple of integers using MATLAB.

SOFTWARE REQUIRED:

Execute MATLAB/Octave Online (GNU Octave, v4.2.1)

SYNTEX:

L = lcm(A,B)

L = lcm(A,B) returns the least common multiple of corresponding elements of arrays A and B. Inputs A and B must contain positive integer elements and must be the same size (or either can be scalar).

QUESTION NO: 1

Find the least common multiple of 8 and 40.

Solution:

MATLAB Program

```
L = lcm(8,40)
```

OUTPUT

L = 40

QUESTION NO: 2

Find LCM of 3/4, 7/3, 11/2, 12/3, 33/4

Solution:

MATLAB Program

```
lcm (sym([3/4, 7/3, 11/2, 12/3, 33/4]))
```

OUTPUT

924

Problems for Practice

1. Find the lowest number which is exactly divisible by 18 and 24.

```
octave:1> lcm(18,24)
ans = 72

727721euit126
```

2. Find the lowest common multiple of 24, 48 .

```
octave:1> lcm(18,24)
ans = 72

727721euit126
```

3. Find the LCM of the following numbers:

(a)4,7,12,84
(b)25,15,36
(c)24,36,40
(d)27,36
(e) 4,8,18

```
octave:6> L=lcm(4,7,12,84)
L = 84
octave:7> L=lcm(25,15,36)
L = 900
octave:8> L=lcm(27,36)
L = 108
octave:9> L=lcm(4,8,18)
L = 72
octave:10> L=lcm(24,36,30)
L = 360
727721euit126
```

4. Find the LCM of 0.6, 9.6 and 0.36.

```
>> L=lcm(sym([0.6,9.6,0.36]))

L =

144/5

>> 727721euit126
```

5. Find the L.C.M. of 0.48, 0.72 and 0.108.

```
>> L=lcm(sym([0.48,0.72,0.108]))

L =

108/25

>> 727721euit126
```

6. Find LCM of 5/8,16/15,2/3

```
>> L=lcm(sym([5/8,16/15,2/3]))

L =

80

>> 727721euit126
```

Conclusion:

MATLAB is an extraordinarily useful tool for all kinds of engineering analysis and design in MATLAB ideas in understandable computer programs and very easy to compute least common multiple.

EX.NO:11	Compute greatest common divisor of two integers using suitable mathematical software
DATE:5.9.22	

OBJECTIVE:

To find greatest common divisor of two integers using MATLAB.

SOFTWARE REQUIRED:

Execute MATLAB/Octave Online (GNU Octave, v4.2.1)

SYNTEX:

G = gcd(A,B)

G = gcd(A,B) returns the greatest common divisor of corresponding elements of arrays A and B. Inputs A and B must contain positive integer elements and must be the same size (or either can be scalar).

QUESTION NO: 1

Find the greatest common divisor of 60 and 90.

Solution:

MATLAB Program

```
G = gcd(60,90)
```

OUTPUT

G = 30

QUESTION NO: 2

Find GCD of 18, 72, 132, 96, 198

Solution:

MATLAB Program

```
G= gcd (18, 72, 132, 96, 198)
```

OUTPUT

6

Problems for Practice

1.Find the greatest common divisor of 625 and 1000.

```
octave:16> G=gcd(625,1000)
G = 125
```

727721euit126

2. Find the greatest common divisor of 12345 and 54321 .

```
octave:17> G=gcd(12345,54321)
G = 3
```

727721euit126

3. Find the gcd (1819, 3587).

```
octave:18> G=gcd(1819,3587)
G = 17
```

727721euit126

4. Find the gcd (512, 320)

```
octave:19> G=gcd(512,320)
G = 64
```

727721euit126

5. Find the gcd (396,504,636).

```
octave:20> G=gcd(396,504,636)
G = 12
```

727721euit126

6. Find the gcd (84, 90,120).

```
octave:21> G=gcd(84,90,120)
G = 6
```

727721euit126

7. Find the GCD of 0.6, 9.6 and 0.36.

```
>> G=gcd(sym([0.6,9.6,0.36]))

G =

3/25

>> 727721euit126
```

8. Find the GCD of 0.48, 0.72 and 0.108.

```
>> G=gcd(sym([0.48,.072,0.108]))

G =

3/250

>> 727721euit126
```

9. Find GCD of 5/8,16/15 ,2/3.

```
>> G=gcd(sym([5/8,16/15,2/3]))

G =

1/120

>> 727721euit126
```

10. Find GCD of 3/4, 7/3, 11/2, 12/3, 33/4

```
>> G=gcd(sym([3/4,7/3,11/2,12/3,33/4]))


G =

1/12


>> 727721euit126
```

Conclusion:

MATLAB is an extraordinarily useful tool for all kinds of engineering analysis and design in MATLAB ideas in understandable computer programs and very easy to compute greatest common divisor.

EX.NO:12	**Compute Quotient and remainder of two integers by division algorithm using suitable mathematical software**
DATE:5.9.22	

OBJECTIVE:

To find Quotient and remainder of two integers using MATLAB.

SOFTWARE REQUIRED:

Execute MATLAB/Octave Online (GNU Octave, v4.2.1)

SYNTEX:

[Q , R] = quorem(A , B , var)
[Q , R] = quorem(A , B)

Description

[Q,R] = quorem(A,B,var) divides A by B and returns the quotient Q and remainder R of the division, such that A = Q*B + R. This syntax regards A and B as polynomials in the variable var.

If A and B are matrices, quorem performs elements-wise division, using var as a variable. It returns the quotient Q and remainder R of the division, such that A = Q.*B + R.

Input Arguments

A — Dividend (numerator)
symbolic integer | polynomial | symbolic vector | symbolic matrix

B — Divisor (denominator)
symbolic integer | polynomial | symbolic vector | symbolic matrix

var — Polynomial variable
symbolic variable

QUESTION NO: 1

Compute the quotient and remainder of the division of these multivariate polynomials x3y4-2xy+5x+1 and xy

Solution:

MATLAB Program

```
syms x y
p1 = x^3*y^4 - 2*x*y + 5*x + 1;
p2 = x*y;
[q, r] = quorem(p1, p2, y)
```

OUTPUT

q =
x^2*y^3 - 2

r =
5*x + 1

QUESTION NO: 2

Compute the quotient and remainder of the division of these univariate polynomials:

$$x5 \text{ and } x3\text{-}2x+5$$

Solution:

MATLAB Program

```
syms x

p = x^3 - 2*x + 5;

[q, r] = quorem(x^5, p)
```

OUTPUT

q =
x^2 + 2

r =
- 5*x^2 + 4*x – 10

QUESTION NO: 3

Compute the quotient and remainder of the division of these integers

$$100000=985q+r$$

Solution:

MATLAB Program

```
[q, r] = quorem(sym(100000), sym(985))
```

OUTPUT

q =
101

r =
515

Problems for Practice

1. Compute the quotient and remainder of the division of these multivariate polynomials x4y4-4xy+6x+2 and x2y2

```
>> syms x y
p1 = x^4*y^4 - 4*x*y + 6*x + 2;
p2 = x^2*y^2;
[q, r] = quorem(p1, p2, y)

q =

x^2*y^2


r =

6*x - 4*x*y + 2

>> 72772leuitl26
```

2. Compute the quotient and remainder of the division of these multivariate polynomials x5y4+4xy+8x+1 and x3y

```
>> syms x y
p1 = x^5*y^4 + 4*x*y + 8*x + 1;
p2 = x^3*y;
[q, r] = quorem(p1, p2, y)

q =

x^2*y^3 + 4/x^2


r =

8*x + 1

>> 72772leuitl26
```

3. Compute the quotient and remainder of the division of these univariate polynomials:x4 and 2x2+3x+7

```
>> syms x
 p = 2*x^2 + 3*x + 7;

 [q, r] = quorem(x^4, p)

q =

x^2/2 - (3*x)/4 - 5/8


r =

(57*x)/8 + 35/8

>> 72772leuit126
```

4. Compute the quotient and remainder of the division of these univariate polynomials:x5 and x3-4x+8

```
>> syms x
 p = x^3 - 4*x + 8;

 [q, r] = quorem(x^5, p)

q =

x^2 + 4


r =

- 8*x^2 + 16*x - 32

>> 72772leuit126
```

5. Compute the quotient and remainder of the division of these integers

54321=12345q+r

```
>> [q, r] = quorem(sym(54321), sym(12345))

q =

4


r =

4941

>> 72772leuit126
```

6.Compute the quotient and remainder of the division of these integers

$$3587=1819q+r$$

```
>> [q, r] = quorem(sym(3587), sym(1819))

q =

1


r =

1768

>> 72772leuitl26
```

7. Compute the quotient and remainder of the division of these integers

$$337500=21600q+r$$

```
>> [q, r] = quorem(sym(337500), sym(21600))

q =

15


r =

13500

>> 72772leuitl26
```

8. Compute the quotient and remainder of the division of these integers

$$7469=2464q+r$$

```
>> [q, r] = quorem(sym(7469),sym(2464))

q =

3


r =

77

>> 72772leuitl26
```

9. Compute the quotient and remainder of the division of these integers

$$9888=6060q+r$$

```
>> [q, r] = quorem(sym(9888),sym(6060))

q =

1


r =

3828

>> 72772leuit126
```

10. Compute the quotient and remainder of the division of these integers

$$14038=1529q+r$$

```
>> [q, r] = quorem(sym(14038),sym(1529))

q =

9


r =

277

>> 72772leuit126
```

Conclusion:

MATLAB is an extraordinarily useful tool for all kinds of engineering analysis and design in MATLAB ideas in understandable computer programs and very easy to compute quotient and remainder.