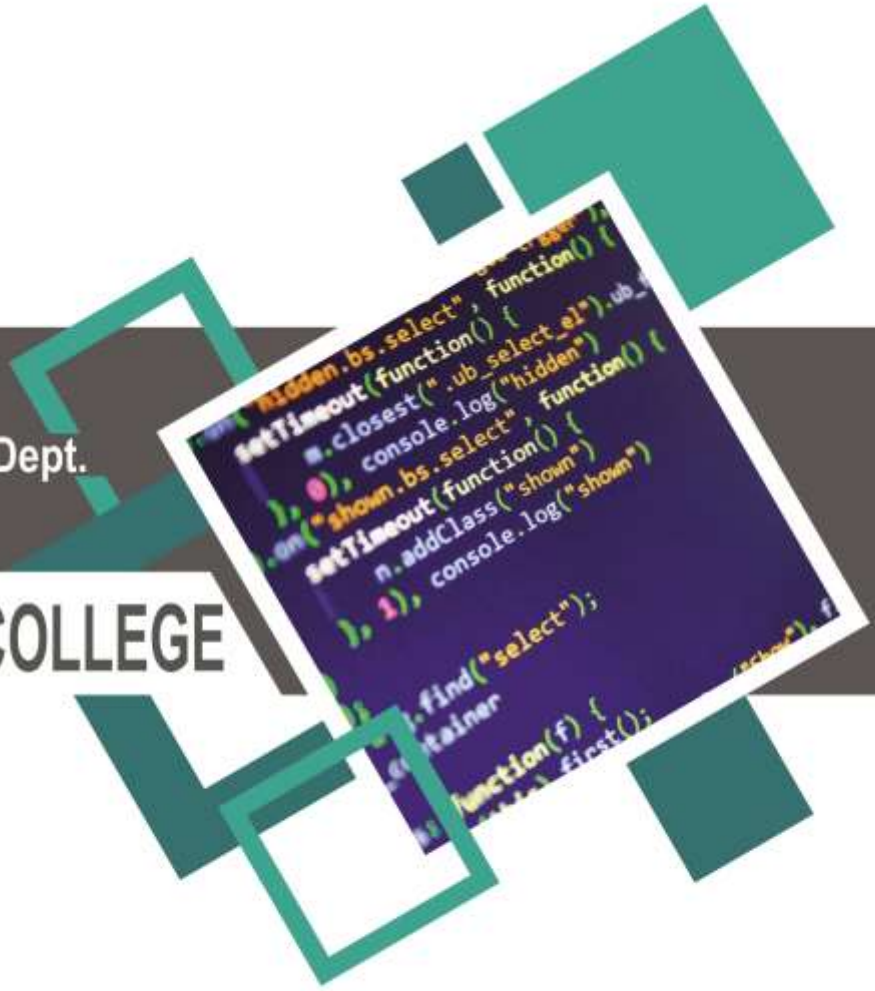
BACHELOR OF COMPUTER APPLICATION LAB MANUAL

2nd Semester

Prepared By

Pure and Applied Science Dept.

Computer Application



MIDNAPORE CITY COLLEGE

DIGITAL LOGIC LABORATORY MANUAL
(Course Code: BCACC3P)

INSTRUCTIONS TO STUDENTS

- Before entering the lab, the student should carry the following things (MANDATORY)

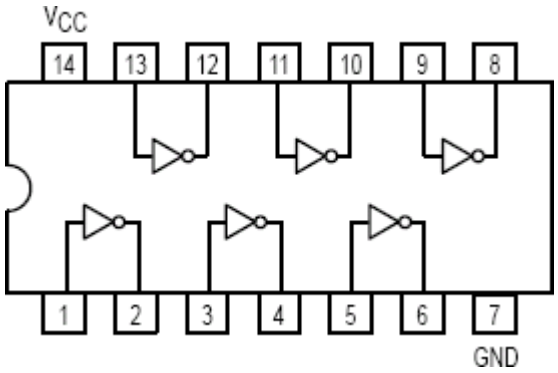
1. Identity card issued by the college.
2. Class notes
3. Lab observation book
4. Lab Manual
5. Lab Record

- Student must sign in and sign out in the register provided when attending the lab session without fail.
- Come to the laboratory in time. Students, who are late more than 10 min., will not be allowed to attend the lab.
- Students need to maintain 80% attendance in lab if not a strict action will be taken.
- All students must follow a Dress Code while in the laboratory.
- Foods, drinks are NOT allowed.
- All bags must be left at the indicated place.
- Refer to the lab staff if you need any help in using the lab.
- Respect the laboratory and its other users.
- Workspace must be kept clean and tidy after experiment is completed.
- Read the Manual carefully before coming to the laboratory and be sure about what you are supposed to do.
- Do the experiments as per the instructions given in the manual.
- Copy all the programs to observation which are taught in class before attending the lab session.
- Students are not supposed to use floppy disks, pen drives without permission of lab- in charge.
- Lab records need to be submitted on or before the date of submission.

CONTENTS

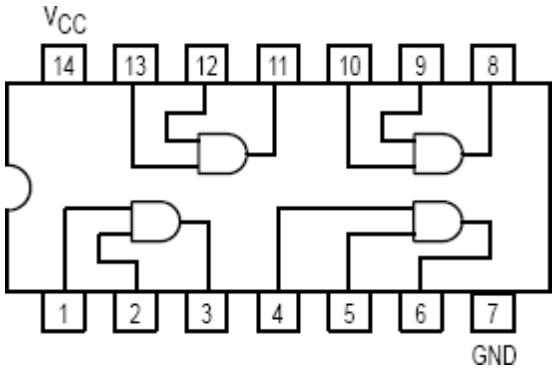
Experiment No

Inverter Gate (NOT Gate) 7404LS



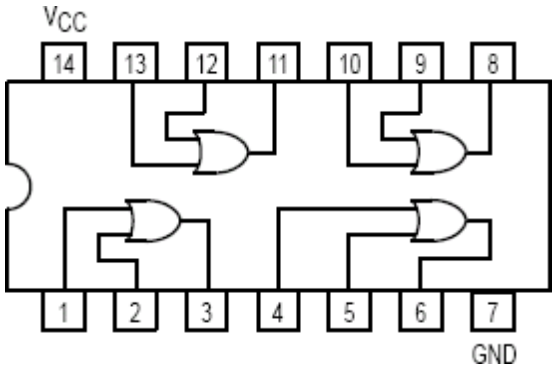
A	O/P	Y1 (V)	Y2 (V)	Y3 (V)	Y4 (V)	Y5 (V)	Y6 (v)
0	1						
1	0						

2-Input AND Gate 7408LS



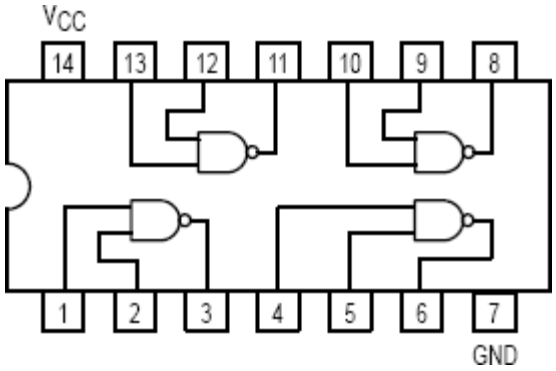
A	B	O/P	Y1 (V)	Y2 (V)	Y3 (V)	Y4 (V)
0	0	0				
0	1	0				
1	0	0				
1	1	1				

2-Input OR Gate 7432LS



A	B	O/P	Y1 (V)	Y2 (V)	Y3 (V)	Y4 (V)
0	0	0				
0	1	0				
1	0	0				
1	1	1				

2-Input NAND Gate 7400LS



A	B	O/P	Y1 (V)	Y2 (V)	Y3 (V)	Y4 (V)
0	0	1				
0	1	0				
1	0	0				
1	1	0				

Experiment No: Date:__/__/

VERIFICATION OF GATES

Aim: - To study and verify the truth table of logic gates

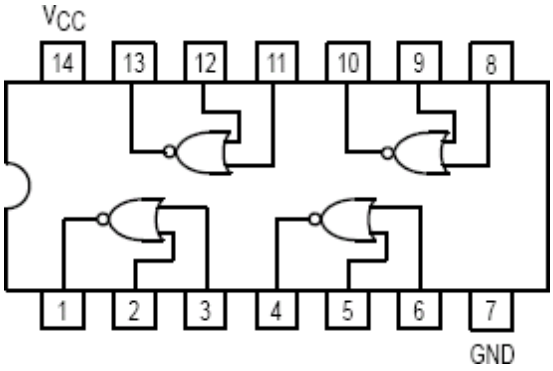
Apparatus Required: -

All the basic gates mention in the fig.

Procedure: -

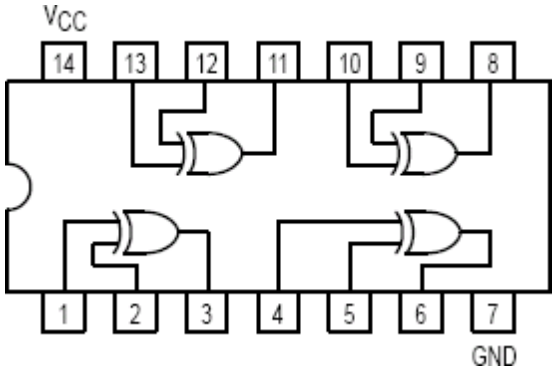
1. **Place the IC-on-IC Trainer Kit.**
2. **Connect V_{cc} and ground to respective pins of IC Trainer Kit.**
3. **Connect the inputs to the input switches provided in the IC Trainer Kit.**
4. **Connect the outputs to the switches of O/P LEDs,**
5. **Apply various combinations of inputs according to the truth table and observe condition of LEDs.**
6. **Disconnect output from the LEDs and note down the corresponding multimeter voltage readings for various combinations of inputs.**

2-Input NOR Gate 7402LS



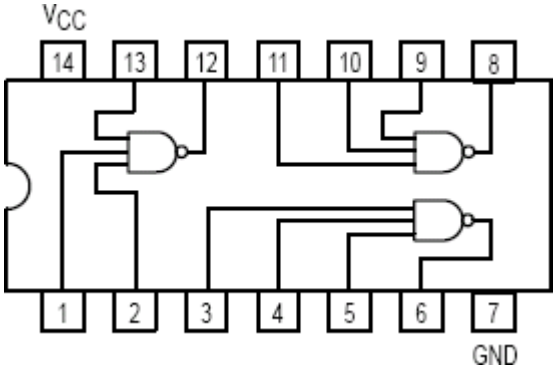
A	B	O/P	Y1 (V)	Y2 (V)	Y3 (V)	Y4 (V)
0	0	1				
0	1	0				
1	0	0				
1	1	0				

2 Input EX-OR Gate 7486LS



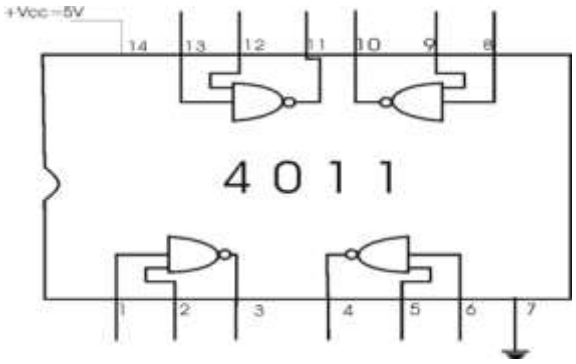
A	B	O/P	Y1 (V)	Y2 (V)	Y3 (V)	Y4 (V)
0	0	0				
0	1	1				
1	0	1				
1	1	0				

3 Input NAND Gate 7410LS



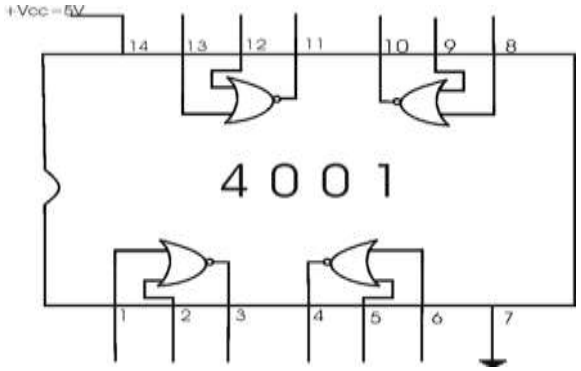
A	B	C	O/P	Y1 (V)	Y2 (V)	Y3 (V)
0	0	0	1			
0	0	1	1			
0	1	0	1			
0	1	1	1			
1	0	0	1			
1	0	1	1			
1	1	0	1			
1	1	1	0			

2-Input NAND Gate CD4011



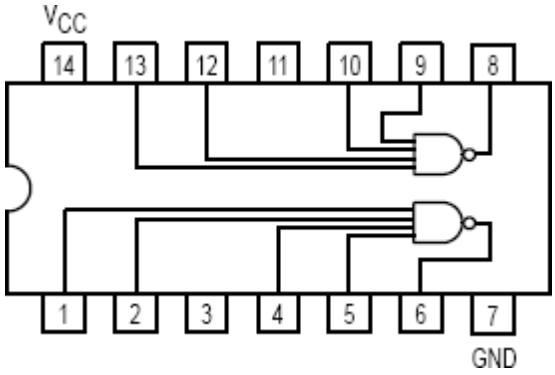
A	B	O/P	Y1 (V)	Y2 (V)	Y3 (V)	Y4 (V)
0	0	1				
0	1	1				
1	0	1				
1	1	0				

2-Input NOR Gate ∈ CD4001



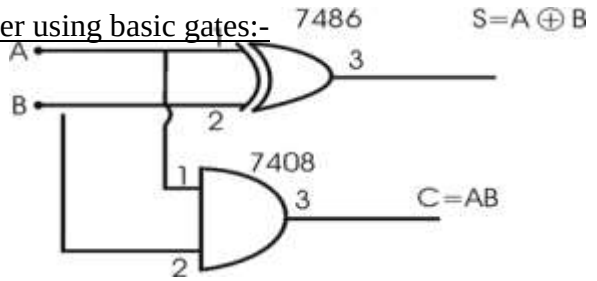
A	B	O/P	Y1 (V)	Y2 (V)	Y3 (V)	Y4 (V)
0	0	1				
0	1	0				
1	0	0				
1	1	0				

4-Input NAND Gate 7420LS

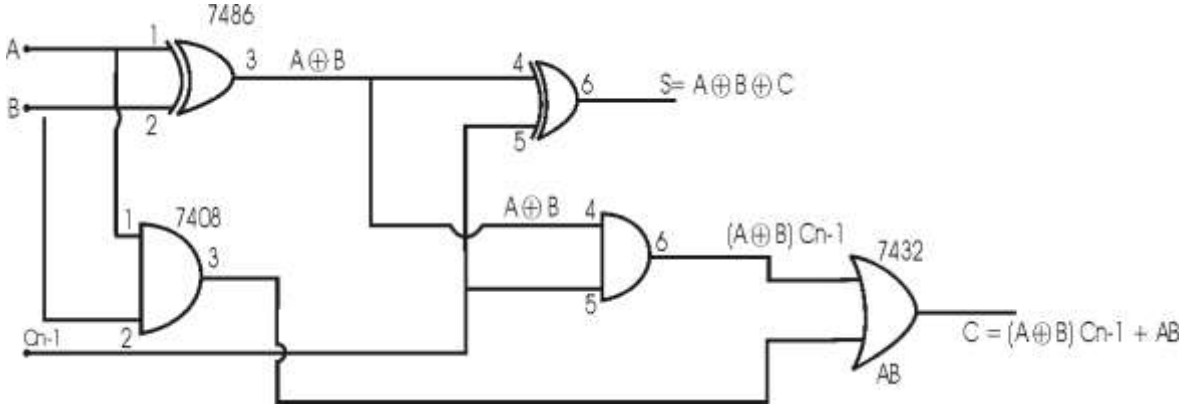


A	B	C	D	O/P	Y1 (V)	Y2 (V)	Y3 (V)
0	0	0	0	1			
0	0	0	1	1			
0	0	1	0	1			
0	0	1	1	1			
0	1	0	0	1			
0	1	0	1	1			
0	1	1	0	1			
0	1	1	1	1			
1	0	0	0	1			
1	0	0	1	1			
1	0	1	0	1			
1	0	1	1	1			
1	1	0	0	1			
1	1	0	1	1			
1	1	1	0	1			
1	1	1	1	0			

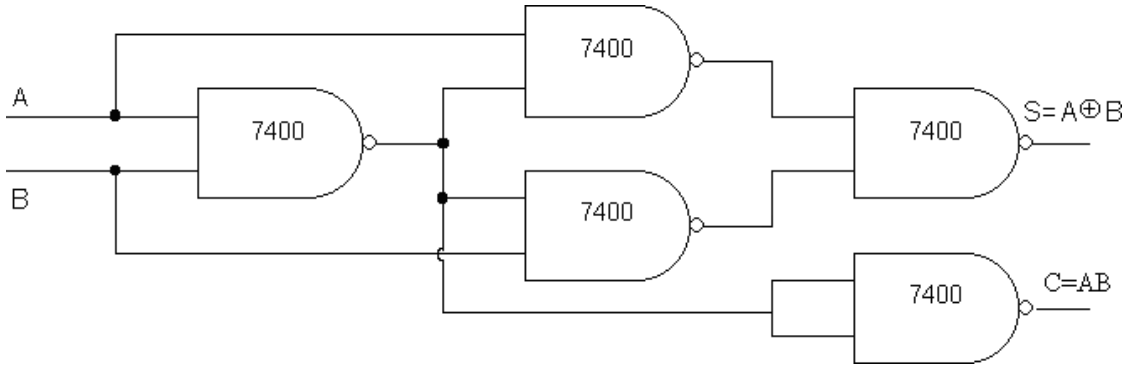
Half Adder using basic gates:-



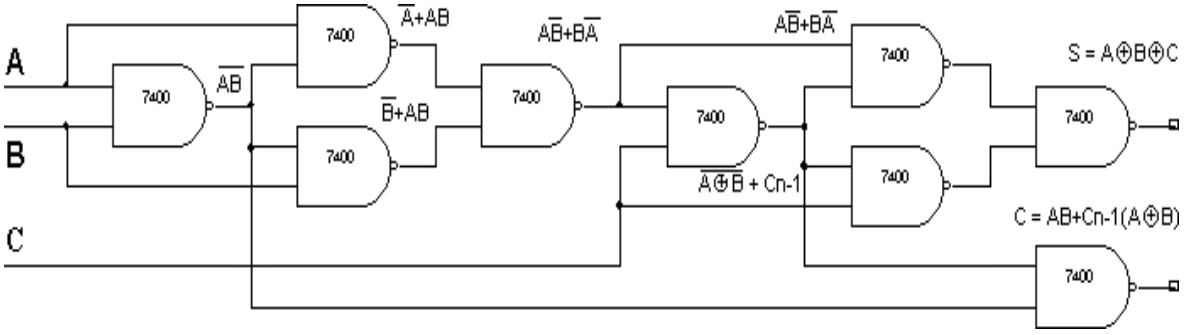
Full Adder using basic gates:-



Half Adder using NAND gates only:-



Full Adder using NAND gates only:-



Experiment No: Date:__/__/

HALF/FULL ADDER & HALF/FULL SUBTRACTOR

Aim: - To realize half/full adder and half/full subtractor.

i. **Using X-OR and basic gates**

ii. **Using only nand gates.**

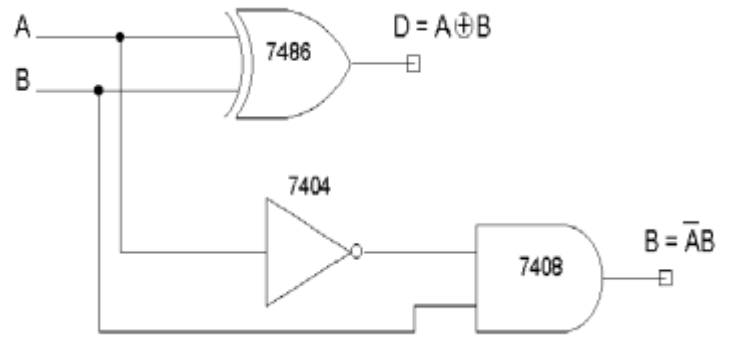
Apparatus Required: -

IC 7486, IC 7432, IC 74 08, IC 7400, etc.

Procedure: -

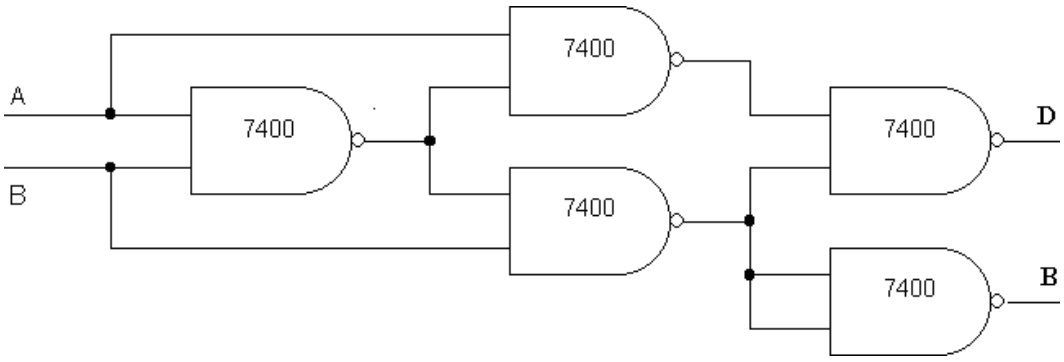
1. **Verify the gates.**
2. **Make the connections as per the circuit diagram.**
3. **Switch on V_{CC} and apply various combinations of input according to the truth table.**
4. **Note down the output readings for half/full adder and half/full subtractor sum/difference and the carry/borrow bit for different combinations of inputs.**

Using X – OR and Basic Gates (a)Half Subtractor

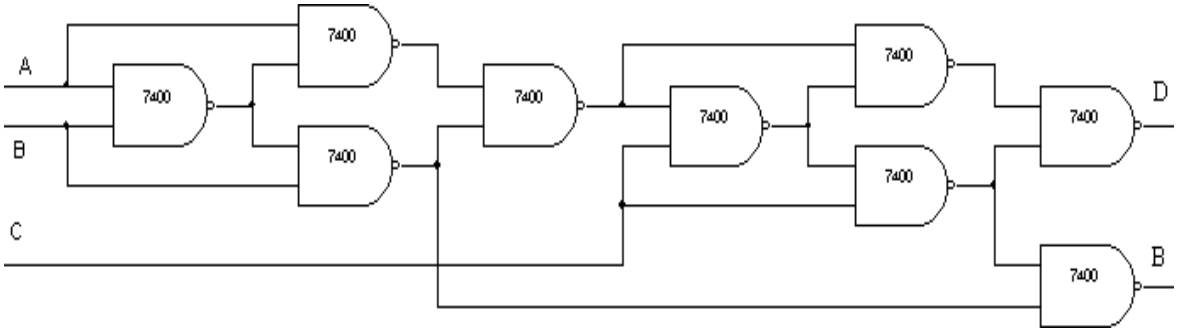


Full Subtractor

(i) Using only NAND gates (a) Half subtractor



(b) Full Subtractor



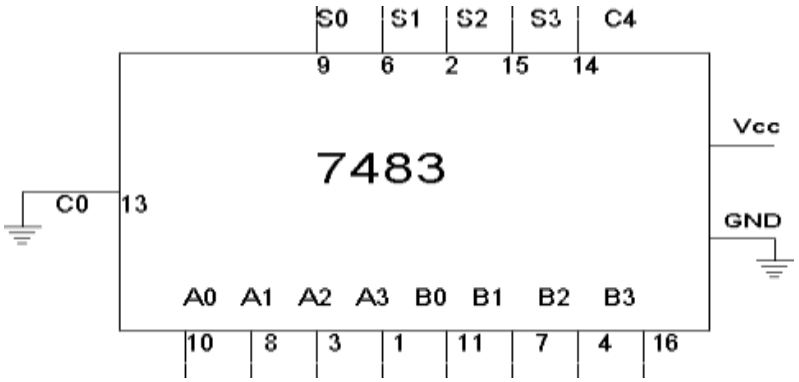
Half Adder					
A	**B**	**S**	**C**	**S(V)**	**C(V)**
	0	**0**	**0**		
0	**1**	**1**	**0**		
1	**0**	**1**	**0**		
1	**1**	**0**	**1**		

Half Subtractor					
A	**B**	**D**	**B**	**D(V)**	**B(V)**
0	**0**	**0**	**0**		
0	**1**	**1**	**1**		
1	**0**	**1**	**0**		
1	**1**	**0**	**0**		

Full Adder						
A	**B**	**Cn-1**	**S**	**C**	**S(V)**	**C(V)**
0	**0**	**0**	**0**	**0**		
0	**0**	**1**	**1**	**0**		
0	**1**	**0**	**1**	**0**		
0	**1**	**1**	**0**	**1**		
1	**0**	**0**	**1**	**0**		
1	**0**	**1**	**0**	**1**		
1	**1**	**0**	**0**	**1**		
1	**1**	**1**	**1**	**1**		

Full Subtractor						
A	**B**	**Cn-1**	**D**	**B**	**D(v)**	**B(v)**
0	**0**	**0**	**0**	**0**		
0	**0**	**1**	**1**	**1**		
0	**1**	**0**	**1**	**1**		
0	**1**	**1**	**0**	**1**		
1	**0**	**0**	**1**	**0**		
1	**0**	**1**	**0**	**0**		
1	**1**	**0**	**0**	**0**		
1	**1**	**1**	**1**	**1**		

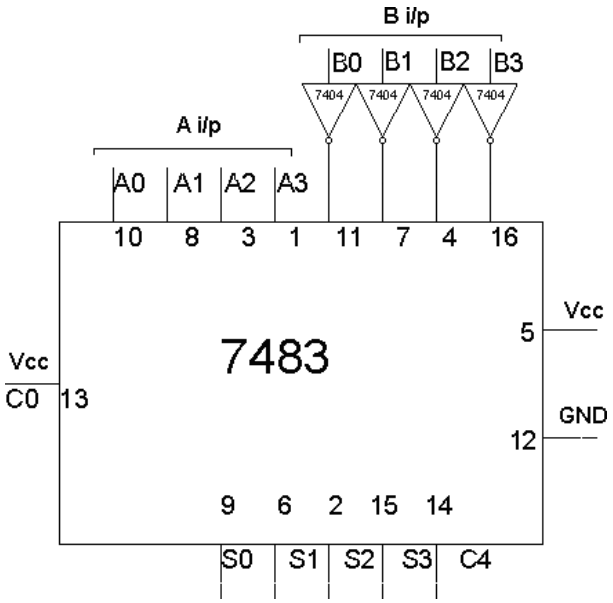
Adder : -



Truth Table: -

A3	A2	A1	A0	B3	B2	B1	B0	C4 (V)	S3(V)	S2(V)	S1(V)	S0(V)
0	0	0	1	0	0	1	0	0	0	0	1	1
0	1	0	1	1	0	1	1	1	1	0	0	0
1	0	1	0	1	0	1	0	1	0	1	0	0
1	1	1	1	1	1	1	1	1	1	1	1	0
0	1	1	1	0	0	1	1	0	1	0	1	0

Subtractor:-



Experiment No: Date:__/__/

PARALLEL ADDER AND SUBTRACTOR USING 7483

Aim: - To realize IC7483 as parallel adder / Subtractor.

Apparatus Required: -

IC 7483, IC 7404, etc.

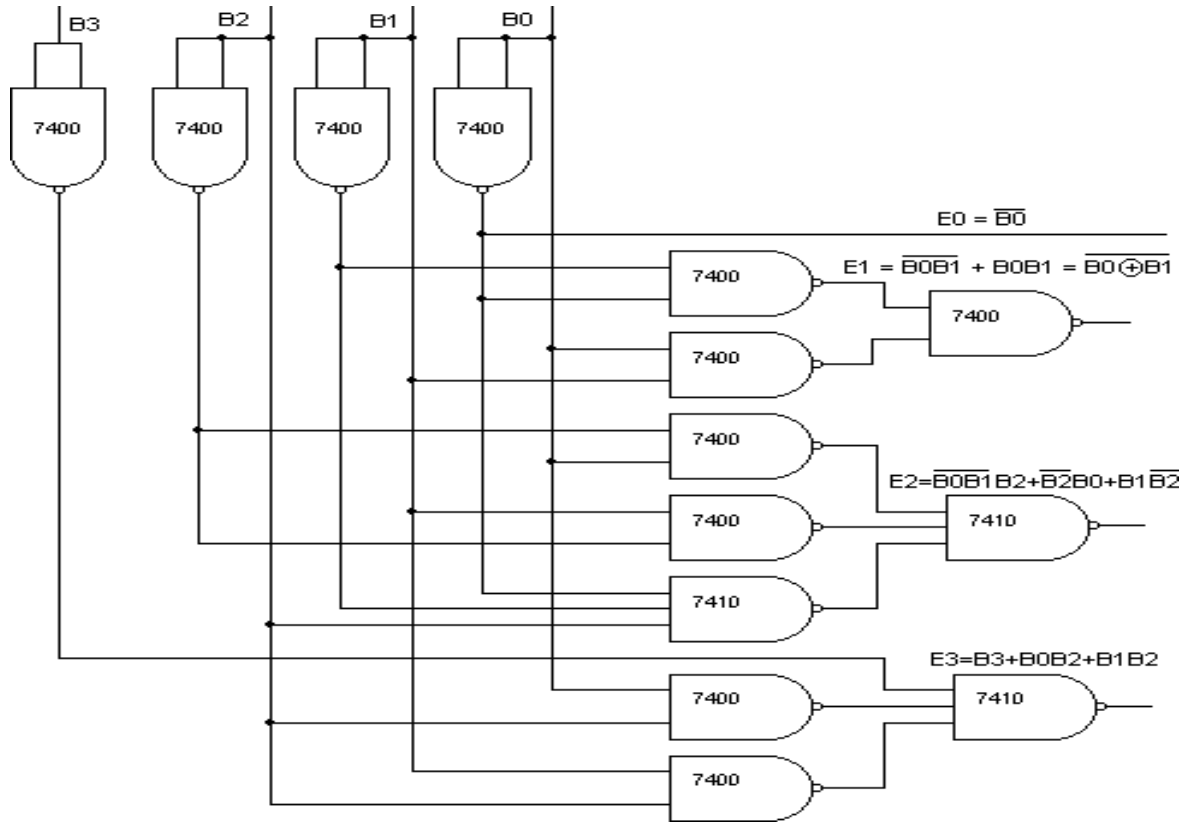
Procedure: -

1. **Apply the inputs to A0 to A3 and B0 to B3.**
2. **Connect C0 to the Ground.**
3. **Check the output sum on the S0 to S3 and also C4.**
4. **For subtraction connect C0 to Vcc, Apply the B input through NOT gate, which gives the complement of B.**
5. **The truth table of adder and Subtractor are noted down.**

Truth Table for Subtractor

A3	A2	A1	A0	B3	B2	B1	B0	C4(V)	S3(V)	S2(V)	S1(V)	S0(V)
0	0	1	0	0	0	0	1	1	0	0	0	1
0	1	0	1	0	0	1	1	1	0	0	1	0
0	0	1	1	0	1	0	1	0	1	1	1	0
1	0	1	0	0	1	1	0	1	0	1	0	0
1	0	0	0	1	1	1	1	0	1	0	0	1
1	0	1	0	0	1	1	0	1	0	1	0	0
1	0	0	0	1	1	1	1	0	1	0	0	1

BCD To Excess-3



Truth Table For Code Conversion: -

Inputs				Outputs			
B3	**B2**	**B1**	**B0**	**E3 (v)**	**E2 (v)**	**E1 (v)**	**E0 (v)**
0	**0**	**0**	**0**	**0**	**0**	**1**	**1**
0	**0**	**0**	**1**	**0**	**1**	**0**	**0**
0	**0**	**1**	**0**	**0**	**1**	**0**	**1**
0	**0**	**1**	**1**	**0**	**1**	**1**	**0**
0	**1**	**0**	**0**	**0**	**1**	**1**	**1**
0	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**1**	**1**	**0**	**1**	**0**	**0**	**1**
0	**1**	**1**	**1**	**1**	**0**	**1**	**0**
1	**0**	**0**	**0**	**1**	**0**	**1**	**1**
1	**0**	**0**	**1**	**1**	**1**	**0**	**0**

Experiment No: Date:__/__/____

BCD to Excess 3 AND Excess 3 to BCD

Aim: - To verify BCD to excess –3 code conversion using NAND gates. To study and verify the truth table of excess-3 to BCD code converter

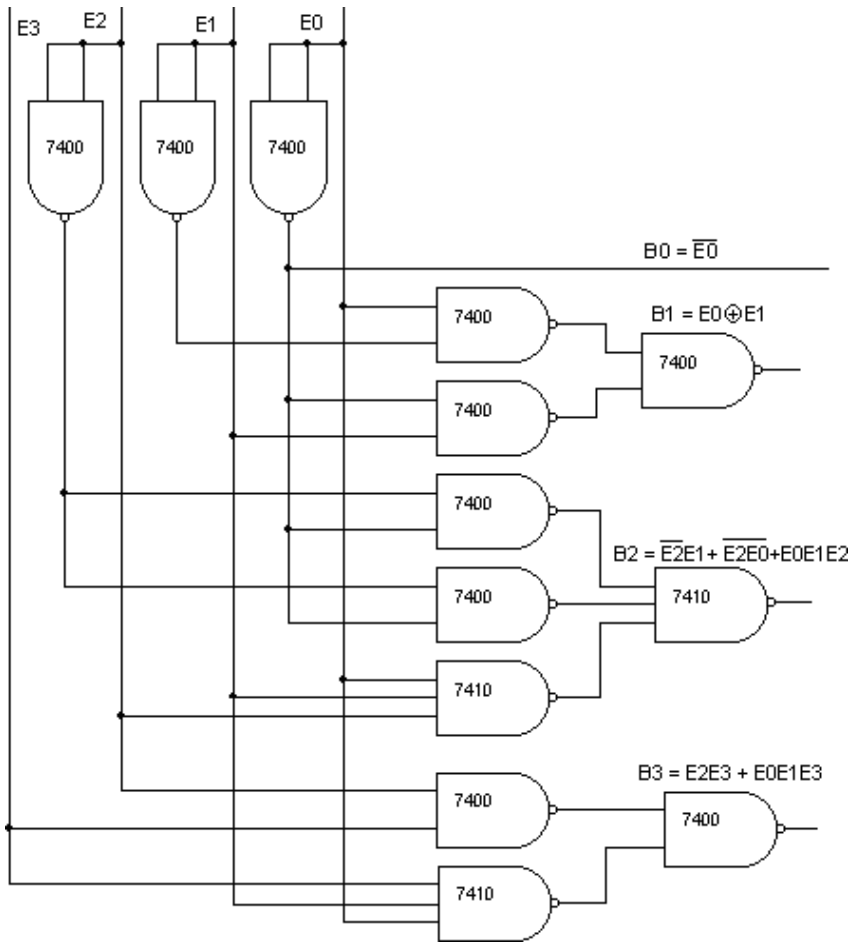
Apparatus Required: -

IC 7400, IC 7404, etc.

Procedure: - (BCD Excess 3 and Vice Versa)

1. **Make the connections as shown in the fig.**
2. **Pin [14] of all IC'S are connected to +5V and pin [7] to the ground.**
3. **The inputs are applied at E3, E2, E1, and E0 and the corresponding outputs at B3, B2, B1, and B0 are taken for excess – 3 to BCD.**
4. **B3, B2, B1, and B0 are the inputs, and the corresponding outputs are E3, E2, E1 and E0 for BCD to excess – 3.**
5. **Repeat the same procedure for other combinations of inputs.**
6. **Truth table is written.**

Excess-3 To BCD :-



Truth Table For Code Conversion: -

Inputs				Outputs			
E3	**E2**	**E1**	**E0**	**B3 (v)**	**B2 (v)**	**B1 (v)**	**B0(v)**
0	**0**	**1**	**1**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**0**	**0**	**1**
0	**1**	**0**	**1**	**0**	**0**	**1**	**0**
0	**1**	**1**	**0**	**0**	**0**	**1**	**1**
0	**1**	**1**	**1**	**0**	**1**	**0**	**0**
1	**0**	**0**	**0**	**0**	**1**	**0**	**1**
1	**0**	**0**	**1**	**0**	**1**	**1**	**0**
1	**0**	**1**	**0**	**0**	**1**	**1**	**1**
1	**0**	**1**	**1**	**1**	**0**	**0**	**0**
1	**1**	**0**	**0**	**1**	**0**	**0**	**1**

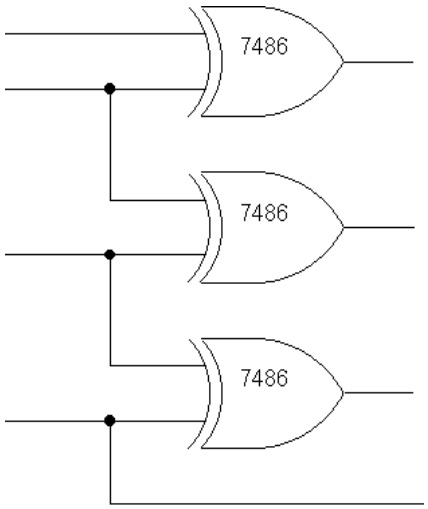
Exercise: -

1. **Obtain the expression for E3, E2, E1 and E0**

2. **Obtain the expression for B3, B2, B1 and B0**

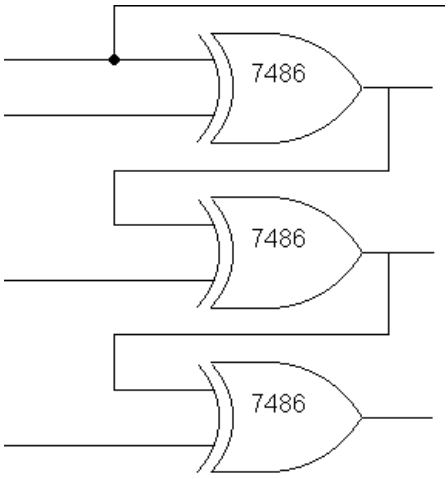
Circuit Diagram: -

Binary To Gray



Using EX-OR gates

Gray To Binary



Using EX-OR gates

Truth Table For Both: -

Inputs				Outputs			
B3	**B2**	**B1**	**B0**	**G3 (V)**	**G2 (V)**	**G1 (V)**	**G0 (V)**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**1**	**0**	**0**	**0**	**1**
0	**0**	**1**	**0**	**0**	**0**	**1**	**1**
0	**0**	**1**	**1**	**0**	**0**	**1**	**0**
0	**1**	**0**	**0**	**0**	**1**	**1**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**1**
0	**1**	**1**	**0**	**0**	**1**	**0**	**1**
0	**1**	**1**	**1**	**0**	**1**	**0**	**0**
1	**0**	**0**	**0**	**1**	**1**	**0**	**0**
1	**0**	**0**	**1**	**1**	**1**	**0**	**1**
1	**0**	**1**	**0**	**1**	**1**	**1**	**1**
1	**0**	**1**	**1**	**1**	**1**	**1**	**0**
1	**1**	**0**	**0**	**1**	**0**	**1**	**0**
1	**1**	**0**	**1**	**1**	**0**	**1**	**1**
1	**1**	**1**	**0**	**1**	**0**	**0**	**1**
1	**1**	**1**	**1**	**1**	**0**	**0**	**0**

Experiment No: Date:__/__/____

BINARY TO GRAY AND GRAY TO BINARY

CONVERSION

Aim: - To convert given binary numbers to gray codes.

Apparatus Required: -

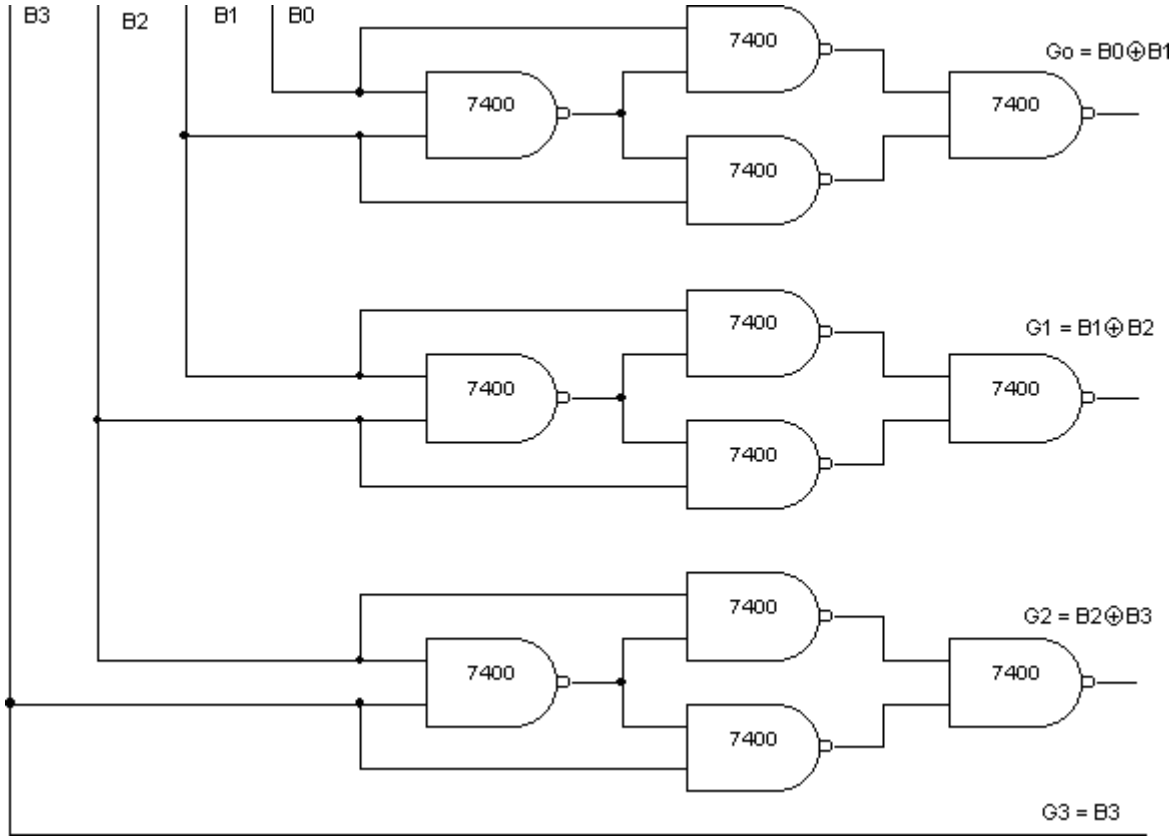
IC 7486, etc

Procedure: -

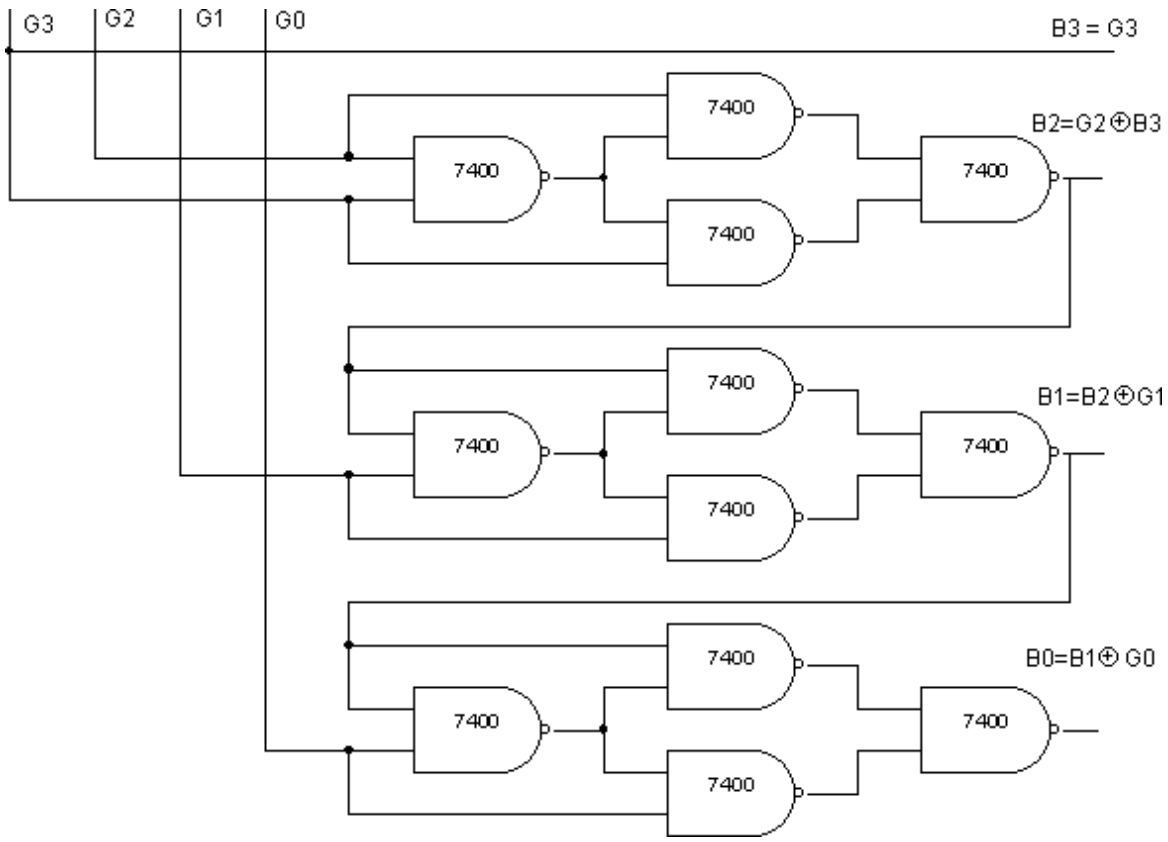
1. **The circuit connections are made as shown in fig.**
2. **Pin (14) is connected to +Vcc and Pin (7) to ground.**
3. **In the case of binary to gray conversion, the inputs B0, B1, B2 and B3 are given at respective pins and outputs G0, G1, G2, G3 are taken for all the 16 combinations of the input.**
4. **In the case of gray to binary conversion, the inputs G0, G1, G2 and G3 are given at respective pins and outputs B0, B1, B2, and B3 are taken for all**

   the 16 combinations of inputs.
5. **The values of the outputs are tabulated.**

Using Nand Gates
Only: - Binary To Gra



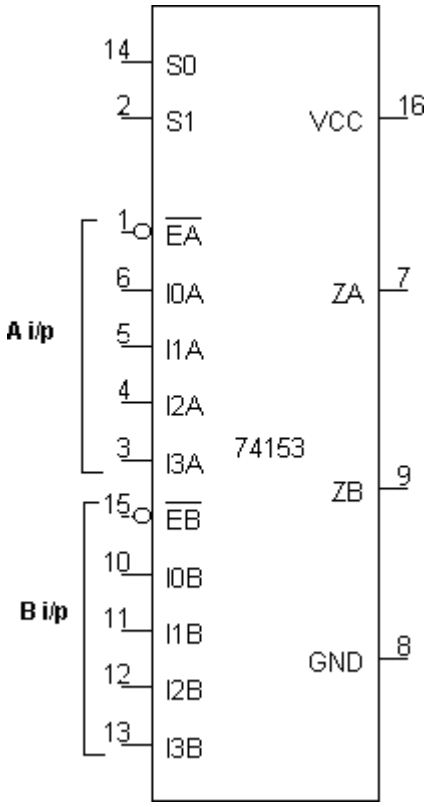
Gray Code



Truth Table For Both: -

Inputs				Outputs			
B3	**B2**	**B1**	**B0**	**G3 (V)**	**G2 (V)**	**G1 (V)**	**G0 (V)**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**1**	**0**	**0**	**0**	**1**
0	**0**	**1**	**0**	**0**	**0**	**1**	**1**
0	**0**	**1**	**1**	**0**	**0**	**1**	**0**
0	**1**	**0**	**0**	**0**	**1**	**1**	**0**
0	**1**	**0**	**1**	**0**	**1**	**1**	**1**
0	**1**	**1**	**0**	**0**	**1**	**0**	**1**
0	**1**	**1**	**1**	**0**	**1**	**0**	**0**
1	**0**	**0**	**0**	**1**	**1**	**0**	**0**
1	**0**	**0**	**1**	**1**	**1**	**0**	**1**
1	**0**	**1**	**0**	**1**	**1**	**1**	**1**
1	**0**	**1**	**1**	**1**	**1**	**1**	**0**
1	**1**	**0**	**0**	**1**	**0**	**1**	**0**
1	**1**	**0**	**1**	**1**	**0**	**1**	**1**
1	**1**	**1**	**0**	**1**	**0**	**0**	**1**
1	**1**	**1**	**1**	**1**	**0**	**0**	**0**

Pin Details: -



Truth Table: -

CHANNEL – A							
INPUTS					SELECT LINES		O/P
Ēa	Ioa	I1a	I2a	I3a	S1	S2	Za(v)
1	X	X	X	X	X	X	0
0	0	X	X	X	0	0	0
0	1	X	X	X	0	0	1
0	X	0	X	X	0	1	0
0	X	1	X	X	0	1	1
0	X	X	0	X	1	0	0
0	X	X	1	X	1	0	1
0	X	X	X	0	1	1	0
0	X	X	X	1	1	1	1

CHANNEL – B							
INPUTS					SELECT LINES		O/P
Ēa	Iob	I1b	I2b	I3b	S1	S2	Za(v)
1	X	X	X	X	X	X	0
0	0	X	X	X	0	0	0
0	1	X	X	X	0	0	1
0	X	0	X	X	0	1	0
0	X	1	X	X	0	1	1
0	X	X	0	X	1	0	0
0	X	X	1	X	1	0	1
0	X	X	X	0	1	1	0
0	X	X	X	1	1	1	1

Experiment No: Date: __/__/

MUX/DEMUX USING 74153 & 74139

Aim: - To verify the truth table of multiplexer using 74153 & to verify a demultiplexer using 74139. To study the arithmetic circuits half-adder half Subtractor, full adder and full Subtractor using multiplexer.

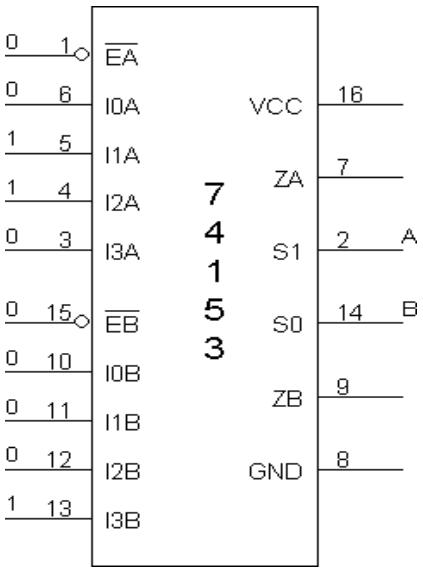
Apparatus Required: -

IC 74153, IC 74139, IC 7404, etc.

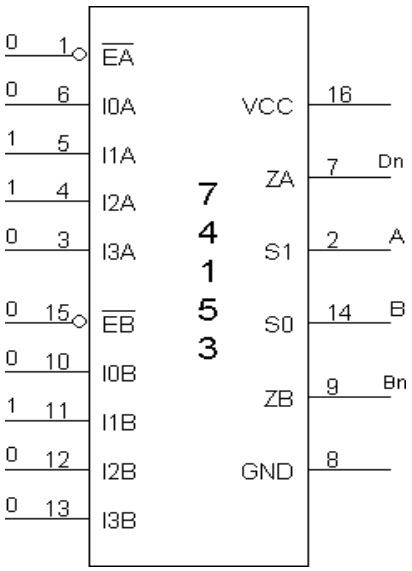
Procedure: - (IC 74153)

1. **The Pin [16] is connected to + Vcc.**
2. **Pin [8] is connected to ground.**
3. **The inputs are applied either to 'A' input or 'B' input.**
4. **If MUX 'A' has to be initialized, Ea is made low and if MUX 'B' has to be initialized, E_b is made low.**
5. **Based on the selection lines one of the inputs will be selected at the output and thus the truth table is verified.**
6. **In case of half adder using MUX, sum and carry is obtained by applying a constant inputs at I_{0a}, I_{1a}, I_{2a}, I_{3a} and I_{0b}, I_{1b}, I_{2b} and I_{3b} and the corresponding values of select lines are changed as per table and the output is taken at Z0a as sum and Z0b as carry.**
7. **In this case, the channels A and B are kept at constant inputs according to the table and the inputs A and B are varied. Making Ea and Eb zero and the output is taken at Za, and Zb.**
8. **In full adder using MUX, the input is applied at Cn-1, An and Bn. According to the table corresponding outputs are taken at Cn and Dn.**

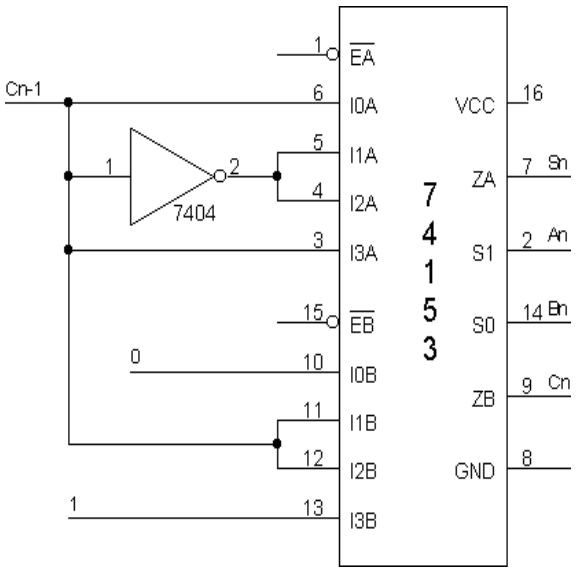
Half Adder Using 74153 –



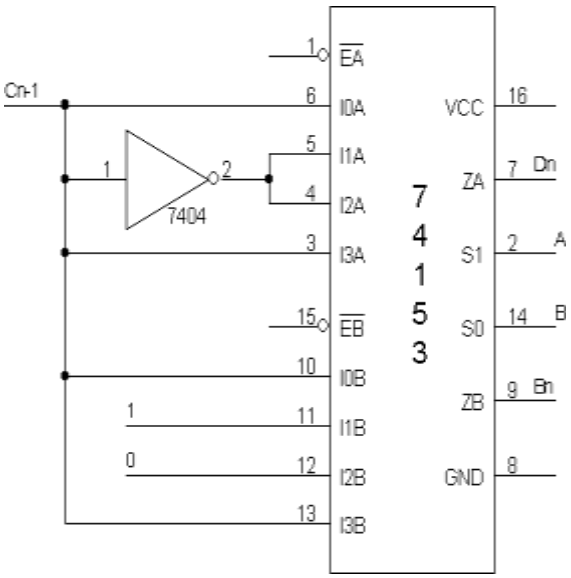
Half Subtractor: -



Full Adder Using 74153: -



Full Subtractor Using 74153: -



Truth Tables: - Same for both Subtractor and adder

Half adder/subtractor			
A	**B**	**Sn/Dn (V)**	**Cn/Bn (V)**
0	**0**		
0	**1**		
1	**0**		
1	**1**		

Full Adder/subtractro				
An	**Bn**	**Cn-1**	**Sn/Dn (V)**	**Cn/Bn (V)**
0	**0**	**0**		
0	**0**	**1**		
0	**1**	**0**		
0	**1**	**1**		
1	**0**	**0**		
1	**0**	**1**		
1	**1**	**0**		
1	**1**	**1**		

Pin Details: -

Truth Table For Demux: -

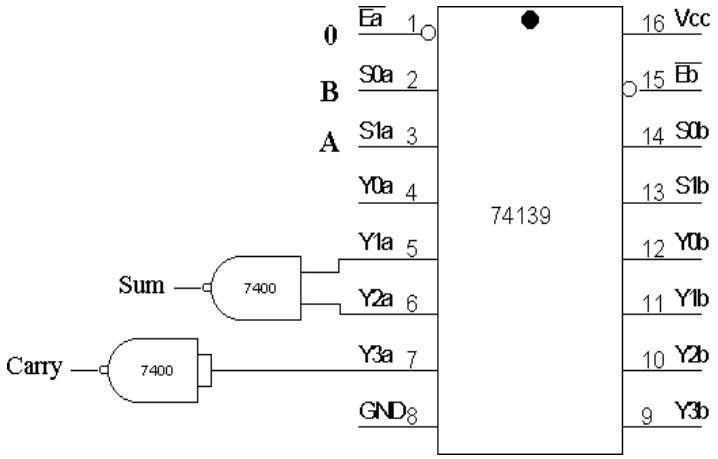
CHANNEL – A						
Inputs			Outputs			
Ēa	S1a	S0a	Y0a	Y1a	Y2a	Y3a
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

CHANNEL – B						
Inputs			Outputs			
Ēb	S1b	S0b	Y0b	Y1b	Y2b	Y3b
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Procedure: - (IC 74139)

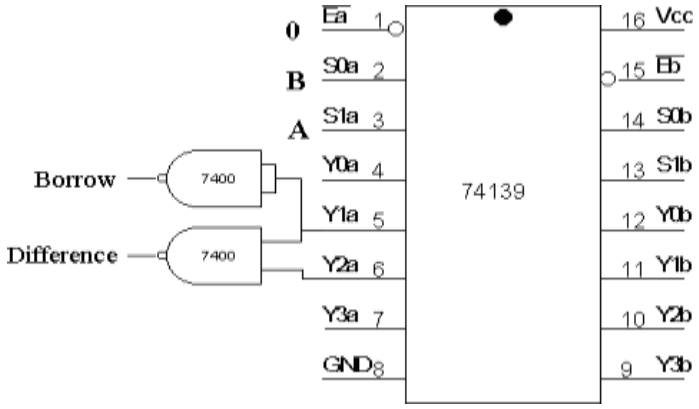
1. **The inputs are applied to either 'a' input or 'b' input**
2. **The demux is activated by making Ea low and Eb low.**
3. **The truth table is verified.**

Half adder



Half Adder			
A	**B**	**Sn (V)**	**Cn (V)**
0	**0**	**0**	**0**
0	**1**	**1**	**0**
1	**0**	**1**	**0**
1	**1**	**0**	**1**

Half subtractor:-

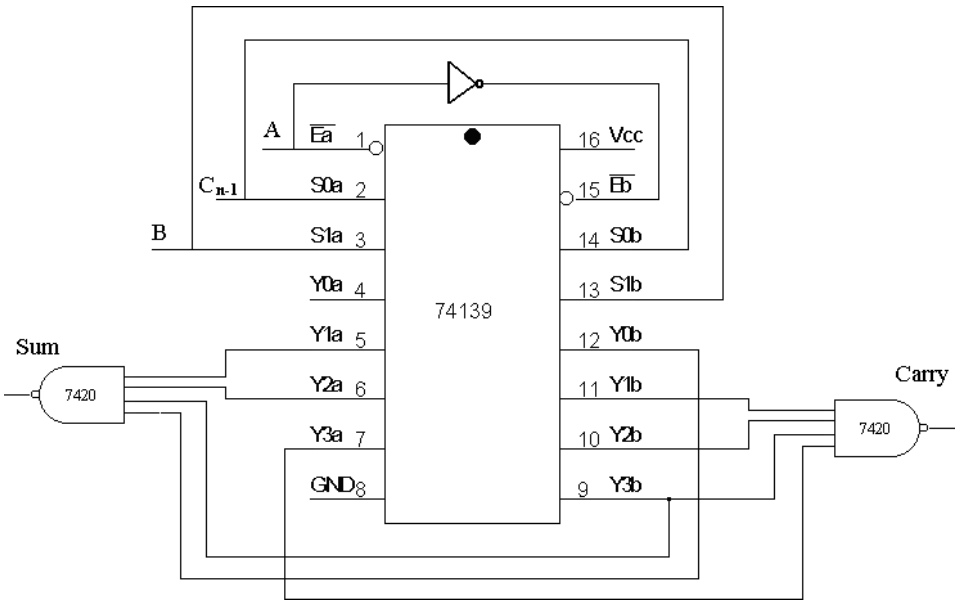


Half Subtractor			
A	**B**	**Dn (V)**	**Bn (V)**
0	**0**	**0**	**0**
0	**1**	**1**	**1**
1	**0**	**1**	**0**
1	**1**	**0**	**0**

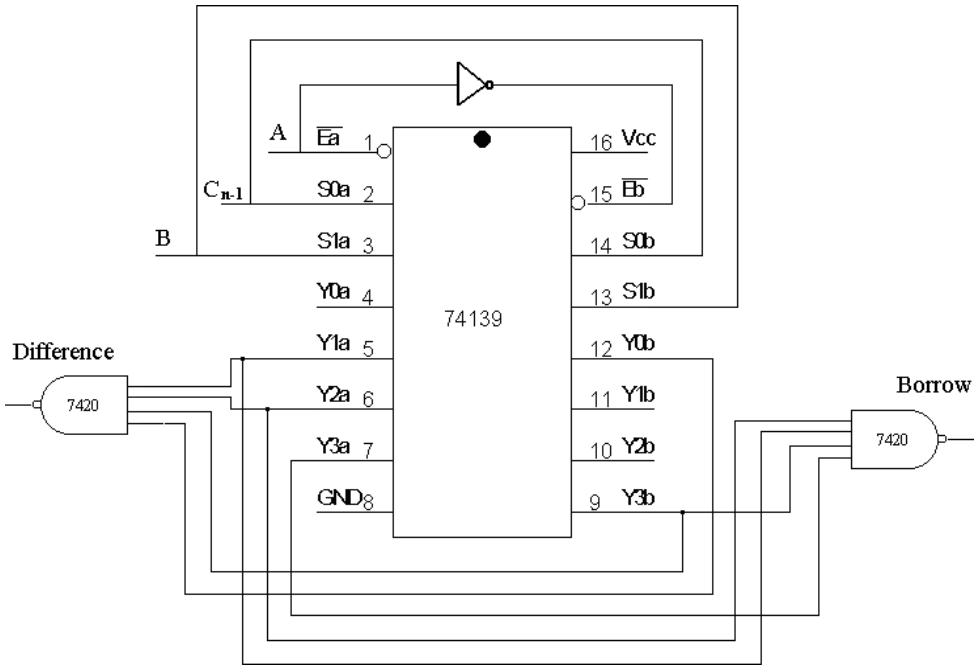
Exercise:-

- **Repeat the experiment to verify**

ChannelB. Full Adder using IC 74139:-



Full subtractor using IC 74139:-

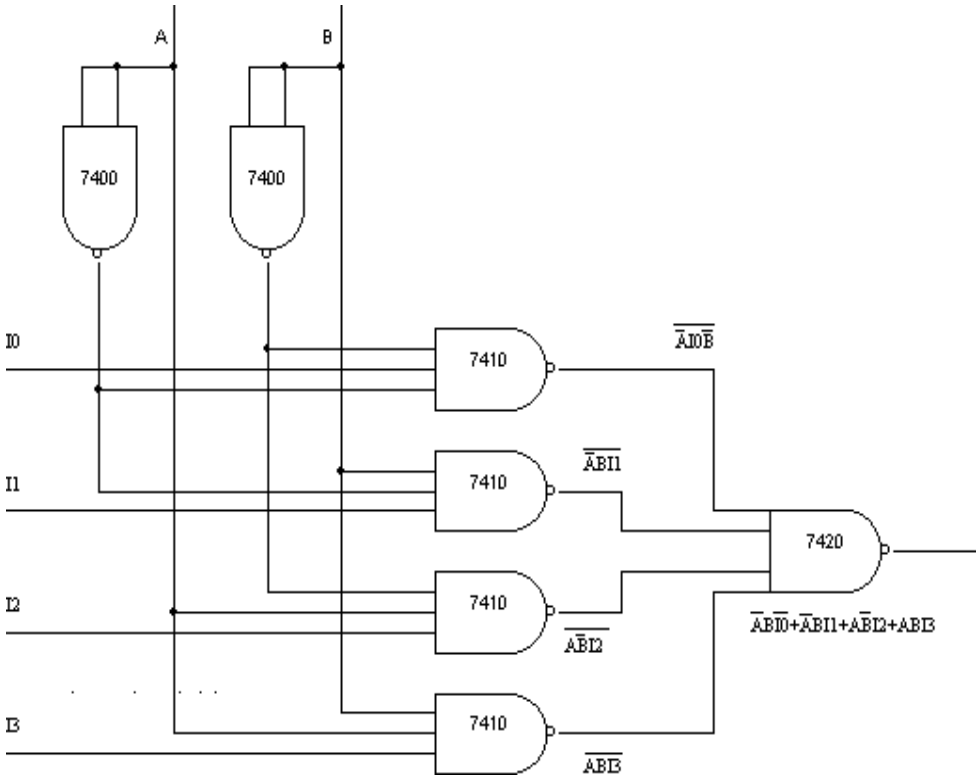


Truth Tables:-

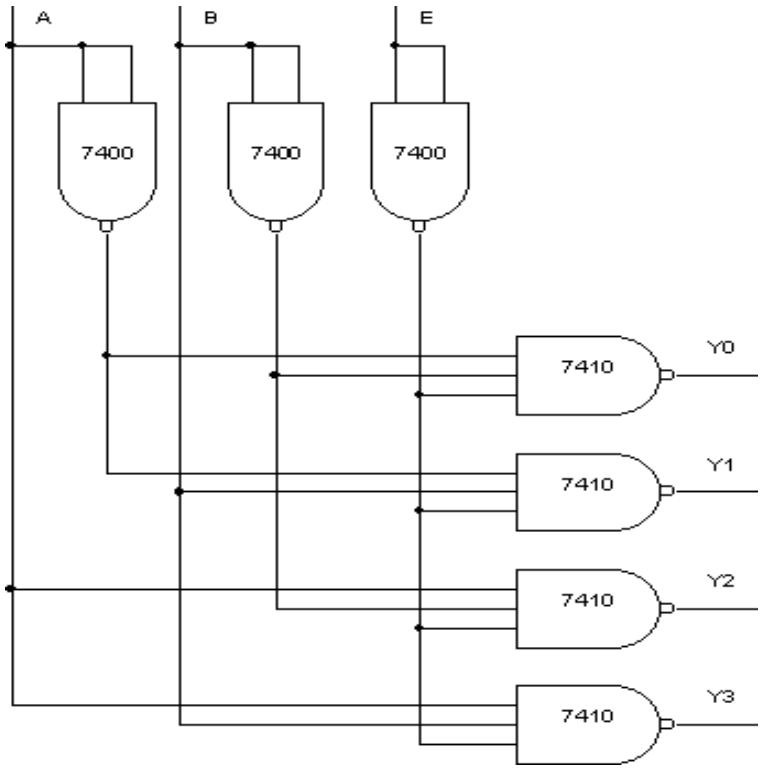
Full Adder				
An	**Bn**	**Cn-1**	**Sn (V)**	**Cn (V)**
0	**0**	**0**		
0	**0**	**1**		
0	**1**	**0**		
0	**1**	**1**		
1	**0**	**0**		
1	**0**	**1**		
1	**1**	**0**		
1	**1**	**1**		

Full Subtractor				
An	**Bn**	**Cn-1**	**Dn (V)**	**Bn (V)**
0	**0**	**0**		
0	**0**	**1**		
0	**1**	**0**		
0	**1**	**1**		
1	**0**	**0**		
1	**0**	**1**		
1	**1**	**0**		
1	**1**	**1**		

MUX USING NAND GATES ONLY: -



DEMUX USING NAND GATES ONLY: -



Experiment No: DATE: __/__/

MUX AND DEMUX USING NAND GATES

AIM: - To verify the truth table of MUX and DEMUX using NAND.

APPARATUS REQUIRED: -

IC 7400, IC 7410, IC 7420, etc.

PROCEDURE: -

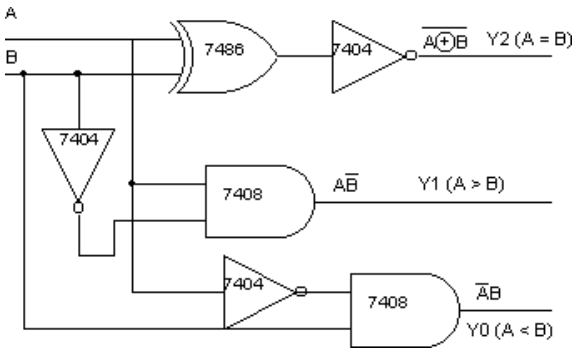
1. **Connections are made as shown in the Circuit diagram.**
2. **Change the values of the inputs as per the truth table and note down the outputs readings using multimeter.**

TRUTH TABLES: -

INPUT						OUPUT
A	B	I0	I1	I2	I3	Y (V)
0	0	0	X	X	X	0
0	0	1	X	X	X	1
0	1	X	0	X	X	0
0	1	X	1	X	X	1
1	0	X	X	0	X	0
1	0	X	X	1	X	1
1	1	X	X	X	0	0
1	1	X	X	X	1	1

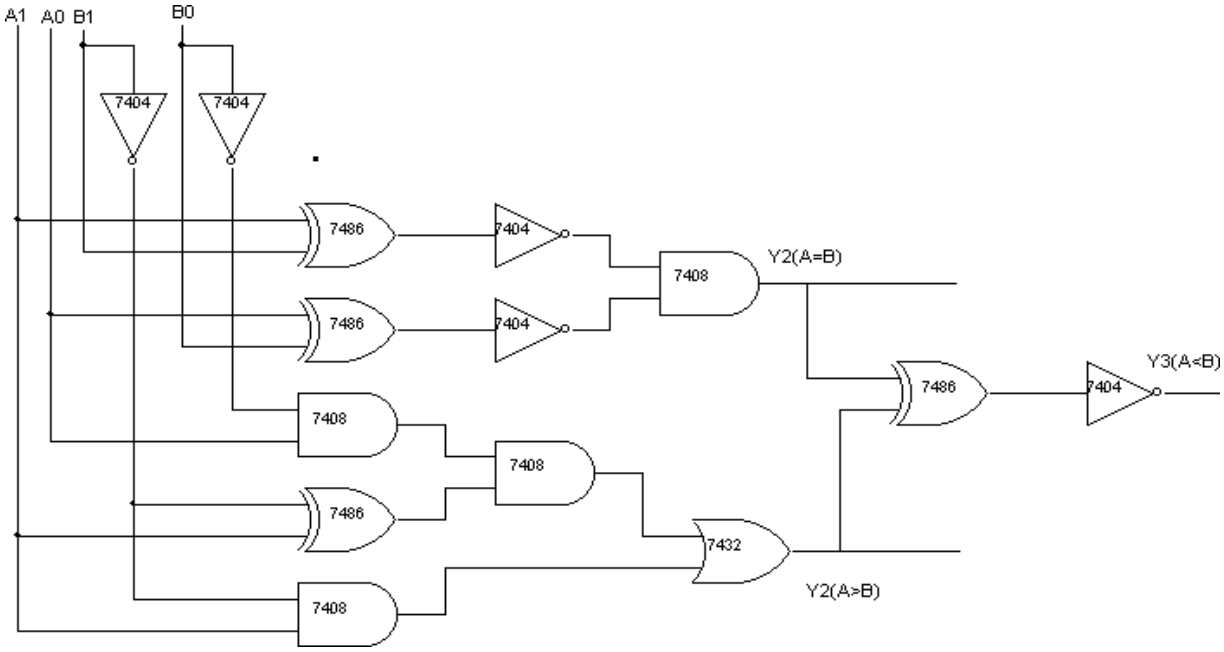
INPUT			OUPUT			
$\bar{E}$	A	B	Y0 (V)	Y1 (V)	Y2 ()	Y3 (V)
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

One Bit Comparator: -



A	B	Y1 (A>B)	Y2 (A = B)	Y3 (A < B)
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Two Bit Comparator: -



Two-Bit Comparator: -

A1	A0	B1	B0	Y1 (A > B)	Y2 (A = B)	Y3 (A < B)
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Experiment No: Date:__/__/____

COMPARATORS

Aim: - To verify the truth t able of one bit and two bit comparators using logic gates.

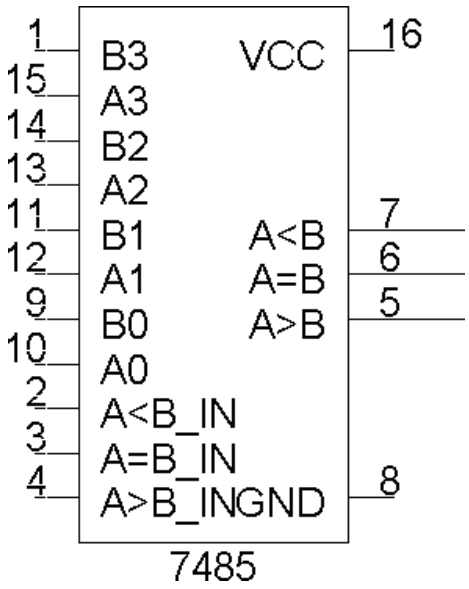
Apparatus Required: -

IC 7486, IC 7404, IC 7408, et c.

Procedure: -

1. **Verify the gates.**
2. **Make the connections as per the circuit diagram.**
3. **Switch on Vcc.**
4. **Applying i/p and Check for the outputs.**
5. **The voltameter readings of outputs are taken and tabulated in tabular column.**
6. **The o/p are verified.**

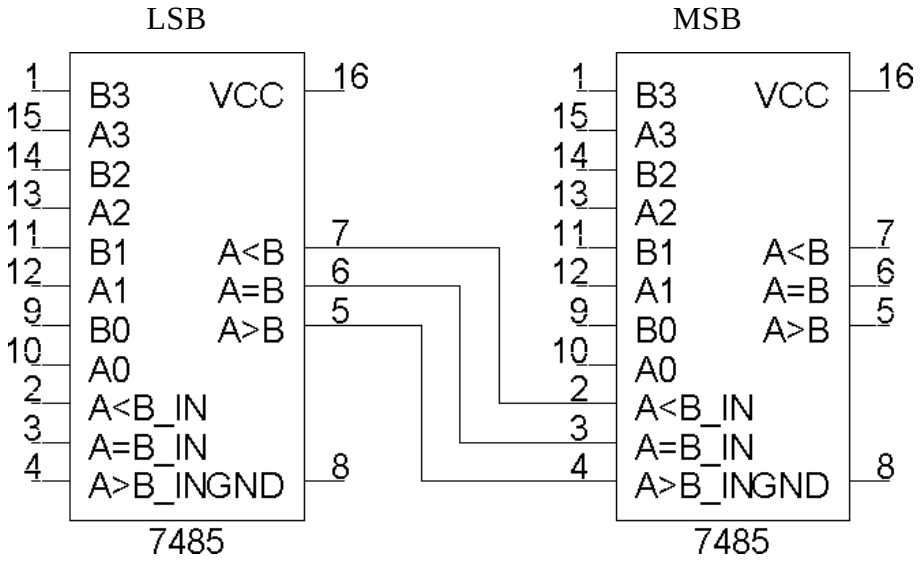
2-bit Comparator



Tabular Coloumn For 8-Bit Comparator: -

A_3 B_3	**A_2 B_2**	**A_1 B_1**	**A_0 B_0**	**A>B**	**A=B**	**A<B**	**A>B**	**A=B**	**A<B**
$A_3>B_3$	**X**	**X**	**X**	**X**	**X**	**X**			
$A_3<B_3$	**X**	**X**	**X**	**X**	**X**	**X**			
$A_3=B_3$	**$A_2>B_2$**	**X**	**X**	**X**	**X**	**X**			
$A_3=B_3$	**$A_2<B_2$**	**X**	**X**	**X**	**X**	**X**			
$A_3=B_3$	**$A_2=B_2$**	**$A_1>B_1$**	**X**	**X**	**X**	**X**			
$A_3=B_3$	**$A_2=B_2$**	**$A_1<B_1$**	**X**	**X**	**X**	**X**			
$A_3=B_3$	**$A_2=B_2$**	**$A_1=B_1$**	**$A_0>B_0$**	**X**	**X**	**X**			
$A_3=B_3$	**$A_2=B_2$**	**$A_1=B_1$**	**$A_0<B_0$**	**X**	**X**	**X**			
$A_3=B_3$	**$A_2=B_2$**	**$A_1=B_1$**	**$A_0=B_0$**	**1**	**0**	**0**			
$A_3=B_3$	**$A_2=B_2$**	**$A_1=B_1$**	**$A_0=B_0$**	**0**	**1**	**0**			
$A_3=B_3$	**$A_2=B_2$**	**$A_1=B_1$**	**$A_0=B_0$**	**0**	**0**	**1**			

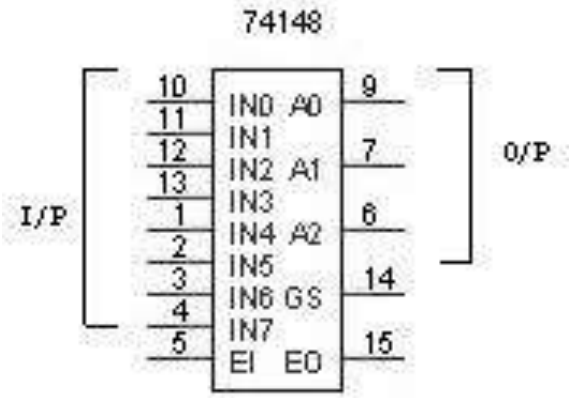
8-Bit Comparator: -



Exercise:-

- **Write the truth table for 8-bit comparator and verify the same for the above circuit.**

PIN DETAILS:-



TRUTH TABLE:-

E_n	A	B	C	D	E	F	G	H	Q_2(V)		Q_1(V)		Q_0(V)		E_S(V)		E_O(V)	
1	X	X	X	X	X	X	X	X	1		1		1		1		1	
0	0	1	1	1	1	1	1	1	1		1		1		0		1	
0	X	0	1	1	1	1	1	1	1		1		0		0		1	
0	0	X	0	1	1	1	1	1	1		0		1		0		1	
0	0	0	X	0	1	1	1	1	1		0		0		0		1	
0	0	0	0	X	0	1	1	1	0		1		1		0		1	
0	0	0	0	0	X	0	1	1	0		1		0		0		1	
0	0	0	0	0	0	X	0	1	0		0		1		0		1	
0	0	0	0	0	0	0	0	0	0		0		0		0		1	
0	1	1	1	1	1	1	1	1	1		1		1		1		0	

Experiment No: DATE:__/__/____

ENCODER & DECODER

AIM:-To convert a given octal input to the binary output and to study the LED display using 7447 7-segment decoder/ driver.

APPARATUS REQUIRED: -

IC 74148, IC 7447, 7-segment display, etc.

PROCEDURE: - (Encoder)

1. **Connections are made as per circuit diagram.**
2. **The octal inputs are given at the corresponding pins.**
3. **The outputs are verified at the corresponding output pins.**

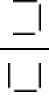
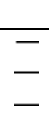
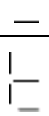
PROCEDURE: - (Decoder)

1. **Connections are made as per the circuit diagram.**
2. **Connect the pins of IC 7447 to the respective pins of the LED display board.**
3. **Give different combinations of the inputs and observe the decimal numbers displayed on the board.**

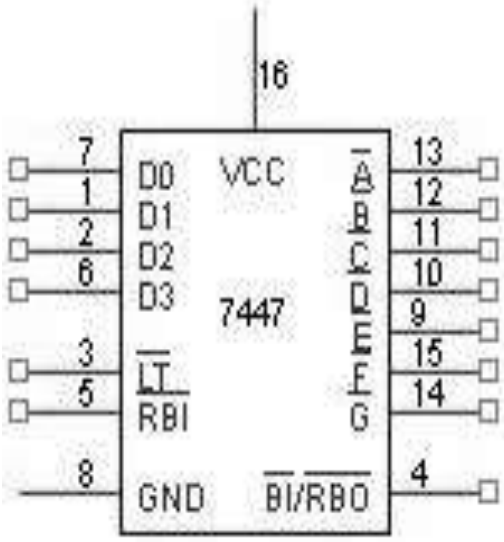
RESULT: -

The given octal numbers are converted into binary numbers. The given data is displayed using &-segment LED decoder.

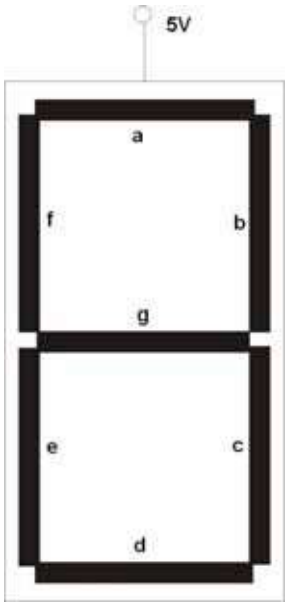
TABULAR COLUMN:-

Q4	Q3	Q2	Q1	O/P	Display	Glowing LEDs
0	0	0	0	0		a,b,c,d,e,f
0	0	0	1	1		b,c
0	0	1	0	2		a,b,d,e,g
0	0	1	1	3		a,b,c,d,g
0	1	0	0	4		b,c,f,g
0	1	0	1	5		a,c,d,f,g
0	1	1	0	6		a.c.d.e.f.g
0	1	1	1	7		a.b.c
1	0	0	0	8		a,b,c,d,e,f,g
1	0	0	1	9		a,b,c,d,f,g
1	0	1	0	10		d,e,g
1	0	1	1	11		c,d,g
1	1	0	0	12		c,d,e
1	1	0	1	13		a,g,d
1	1	1	0	14		d,e,f,g
1	1	1	1	15		blank

PIN DETAILS:-

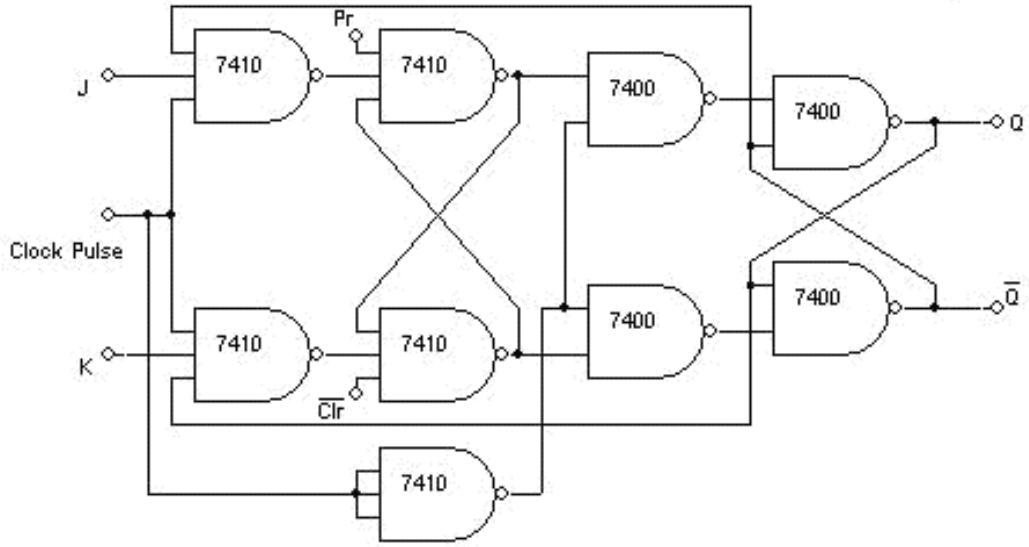


DISPLAY:-

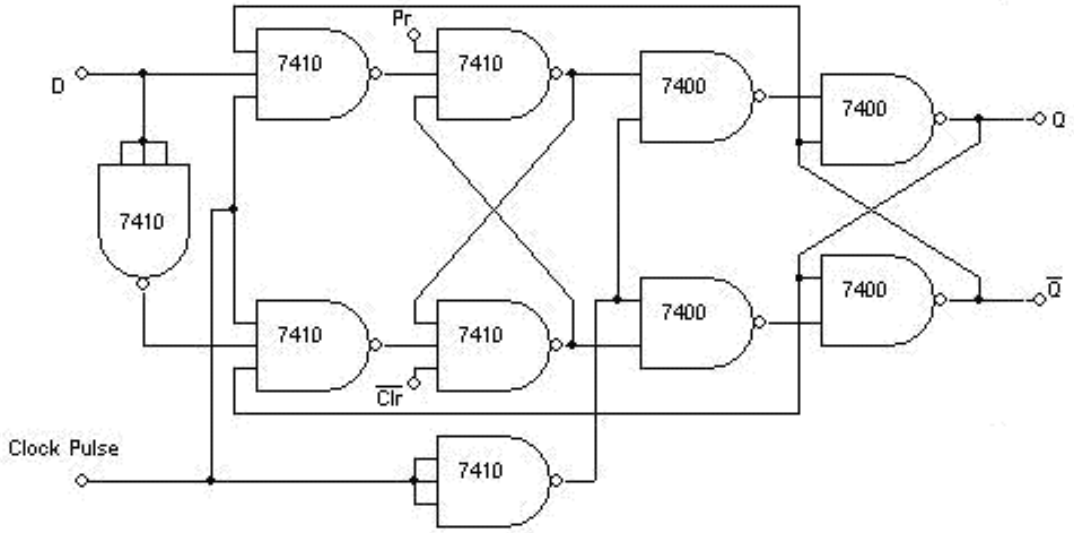


Conclusion:-

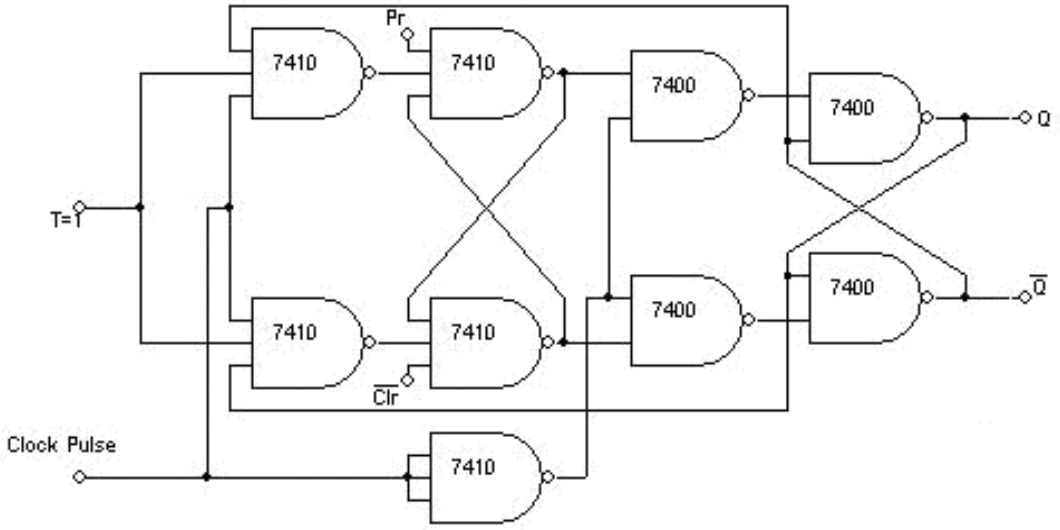
Circuit Diagram: - (Mas ter Slave JK Flip-Flop)



D Flip-Flop:-



T-Flip Flop


Pr
7410
7410
7400
7400
Q
T=1
7410
7410
7400
7400
$\overline{Q}$
$\overline{Clr}$
Clock Pulse
7410

Experiment No: Date:__/__/

FLIP-FLOP

Aim:- Truth table verification of Flip-Flops : (i) JK Master Slave
(ii) D- Type
(iii) T- Type.

Apparatus Required: -
IC 7410, IC 7400, etc.

Procedure: -
1. **Connections are made as per circuit diagram.**
2. **The truth table is verified for various combinations of inputs.**

Truth Table:- (Master Slave JK Flip-Flop)

Preset	Clear	J	K	Clock	Qn+1	$\overline{Qn}$ 1	
0	**1**	**X**	**X**	**X**	**1**	**0**	**Set**
1	**0**	**X**	**X**	**X**	**0**	**1**	**Reset**
1	**1**	**0**	**0**	⎍	**Qn**	$\overline{Qn}$	**No Change**
1	**1**	**0**	**1**	⎍	**0**	**1**	**Reset**
1	**1**	**1**	**0**	⎍	**1**	**0**	**Set**
1	**1**	**1**	**1**	⎍	$\overline{Qn}$	**Qn**	**Toggle**

D Flip-Flop:-

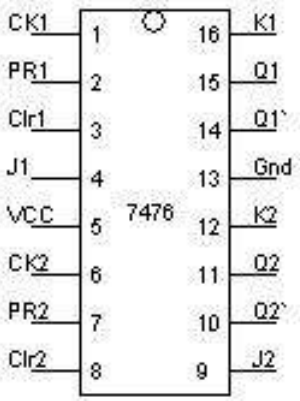
Preset	Clear	D	Clock	Qn+1	$\overline{Qn}$ 1
1	**1**	**0**	⎍	**0**	**1**
1	**1**	**1**	⎍	**1**	**0**

T Flip-Flop:-

Preset	Clear	T	Clock	Qn+1	$\overline{Qn}$ 1
1	**1**	**0**	⎍	**Qn**	$\overline{Qn}$
1	**1**	**1**	⎍	$\overline{Qn}$	**Qn**

Exercise:-
- **Write the timing diagrams for all the above Flip-Flops**

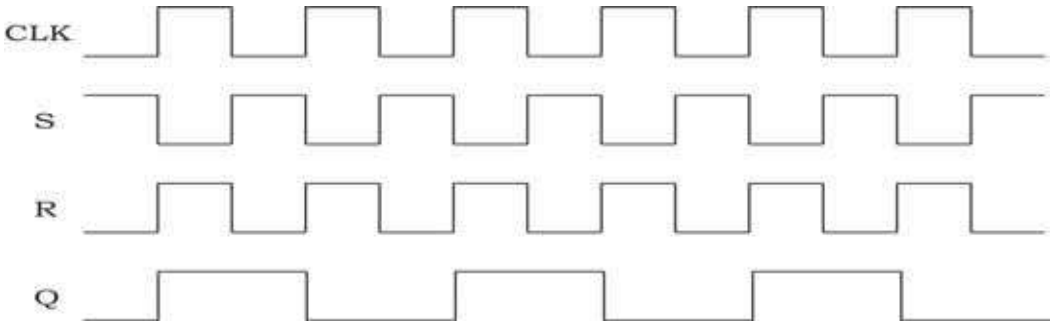
Pin Details: -



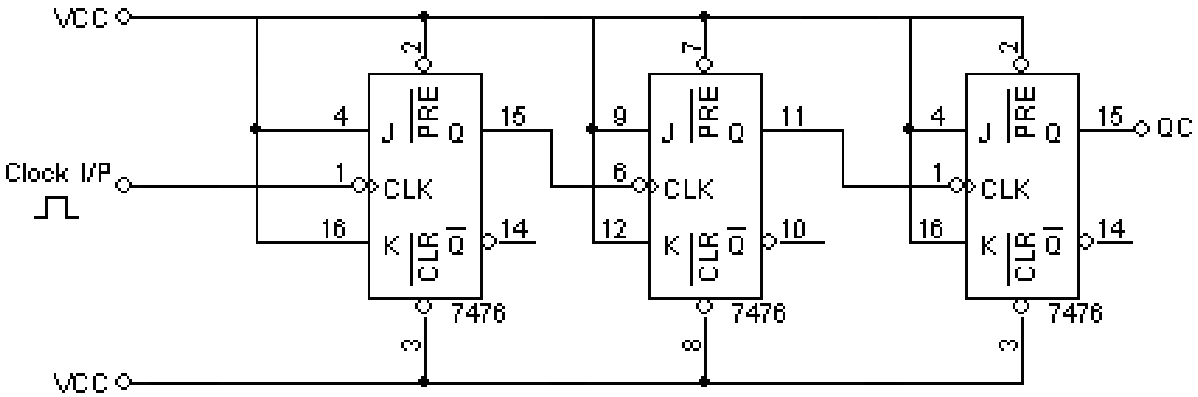
Truth Table:-

Clock	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

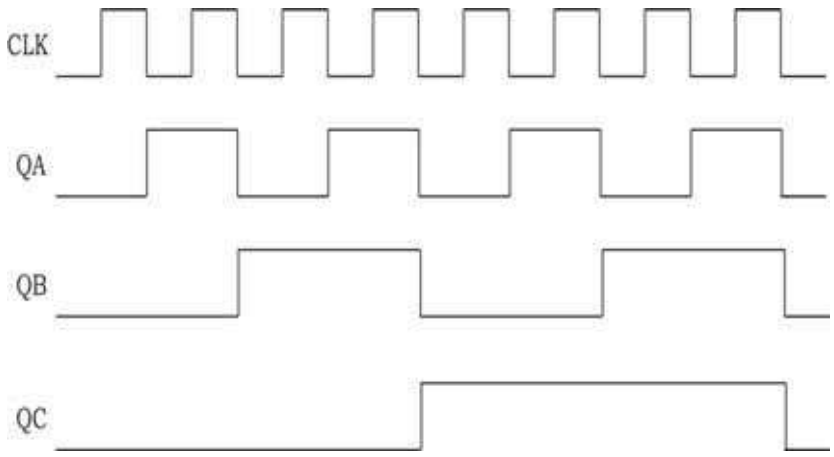
Timing Diagram:-



Circuit Diagram: - 3-Bit Asynchronous Up Counter



3-bit Asynchronous up counter			
Clock	**QC**	**QB**	**QA**
0	**0**	**0**	**0**
1	**0**	**0**	**1**
2	**0**	**1**	**0**
3	**0**	**1**	**1**
4	**1**	**0**	**0**
5	**1**	**0**	**1**
6	**1**	**1**	**0**
7	**1**	**1**	**1**
8	**0**	**0**	**0**



Experiment No: Date:__/__/

COUNTERS

Aim:- Realization of 3-bit counters as a sequential circuit and Mod-N counter design (7476, 7490, 74192, 74193).

Apparatus Required: -

IC 7408, IC 7476, IC 7 490, IC 74192, IC 74193, IC 7400, IC 7416, IC 7432 etc.

Procedure: -

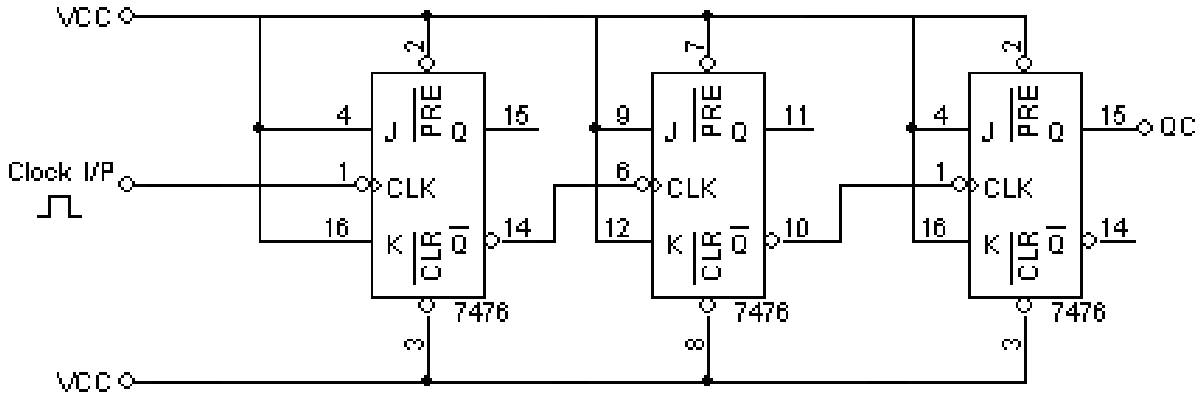
1. **Connections are made as per circuit diagram.**
2. **Clock pulses are applied one by one at the clock I/P and the O/P is observed at QA, QB & QC for IC 7476.**
3. **Truth table is verified.**

Procedure (IC 741 92, IC 74193):-

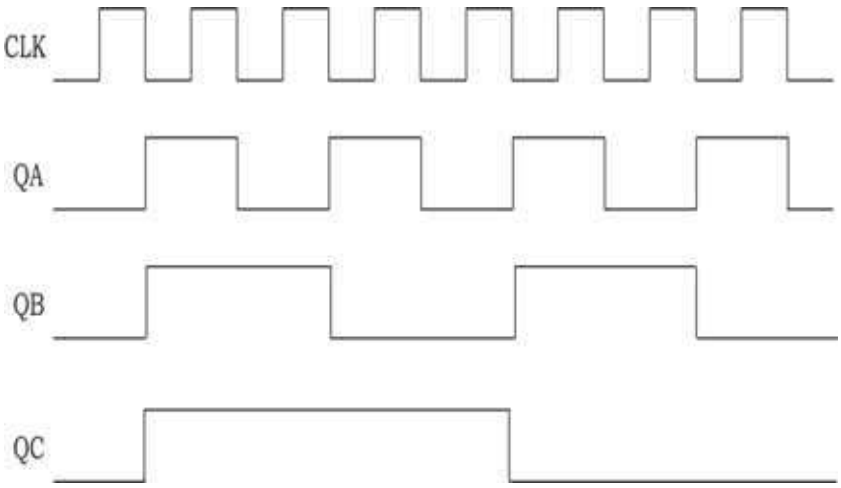
1. **Connections are made as per the circuit diagram except the connection from output of NAND gate to the load input.**
2. **The data (0011) = 3 is made available at the da ta i/ps A, B, C & D respectively.**
3. **The load pin made low so that the data 0011 appears at QD, QC, QB & QA respectively.**
4. **Now connect the output of the NAND gate to the load input.**
5. **Clock pulses are applied to "count up" pin and the truth table is verified.**
6. **Now apply (1100) = 12 for 12 to 5 count er and remaining is same as for 3 to 8 counter.**

7. **The pin diagram of IC 7419 2 is same as that of 74193. 74192 can be configured to count between 0 and 9 in either direction. The starting value can be any number between 0 and 9.**

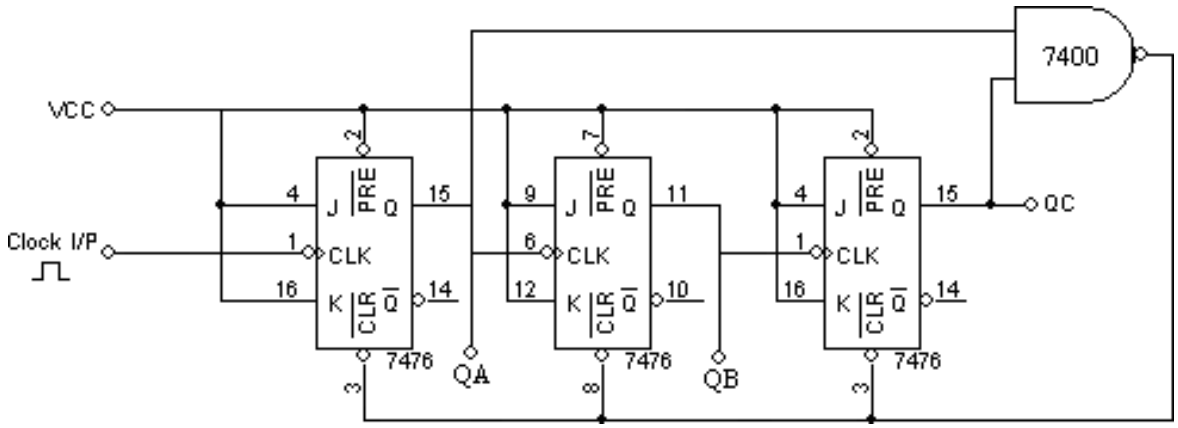
Circuit Diagram: - 3-Bit Asynchronous Down Counter



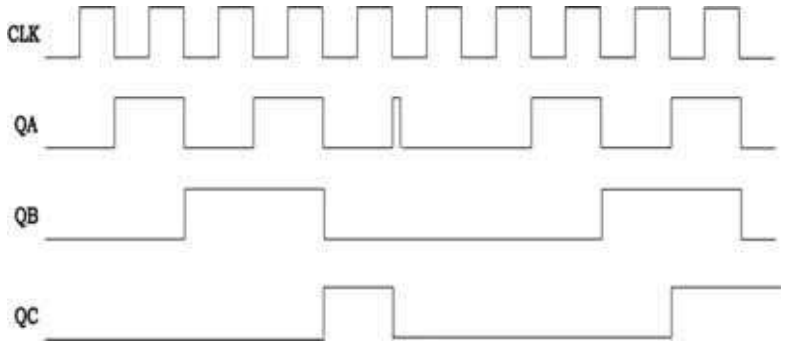
3-bit Asynchronous down counter			
Clock	**QC**	**QB**	**QA**
0	**1**	**1**	**1**
1	**1**	**1**	**0**
2	**1**	**0**	**1**
3	**1**	**0**	**0**
4	**0**	**1**	**1**
5	**0**	**1**	**0**
6	**0**	**0**	**1**
7	**0**	**0**	**0**
8	**1**	**1**	**1**
9	**1**	**1**	**0**



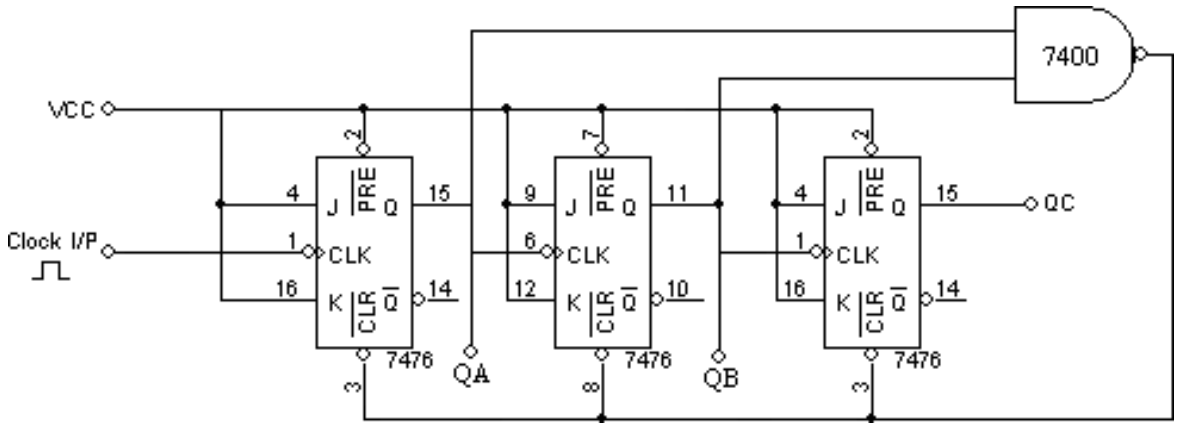
Mod 5 Asynchronous Counter: -



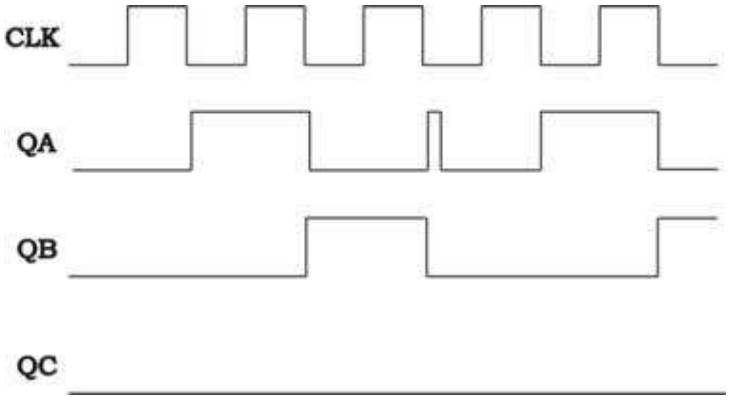
MOD 5 Asynchronous counter			
Clock	**QC**	**QB**	**QA**
0	**0**	**0**	**0**
1	**0**	**0**	**1**
2	**0**	**1**	**0**
3	**0**	**1**	**1**
4	**1**	**0**	**0**
5	**0**	**0**	**0**



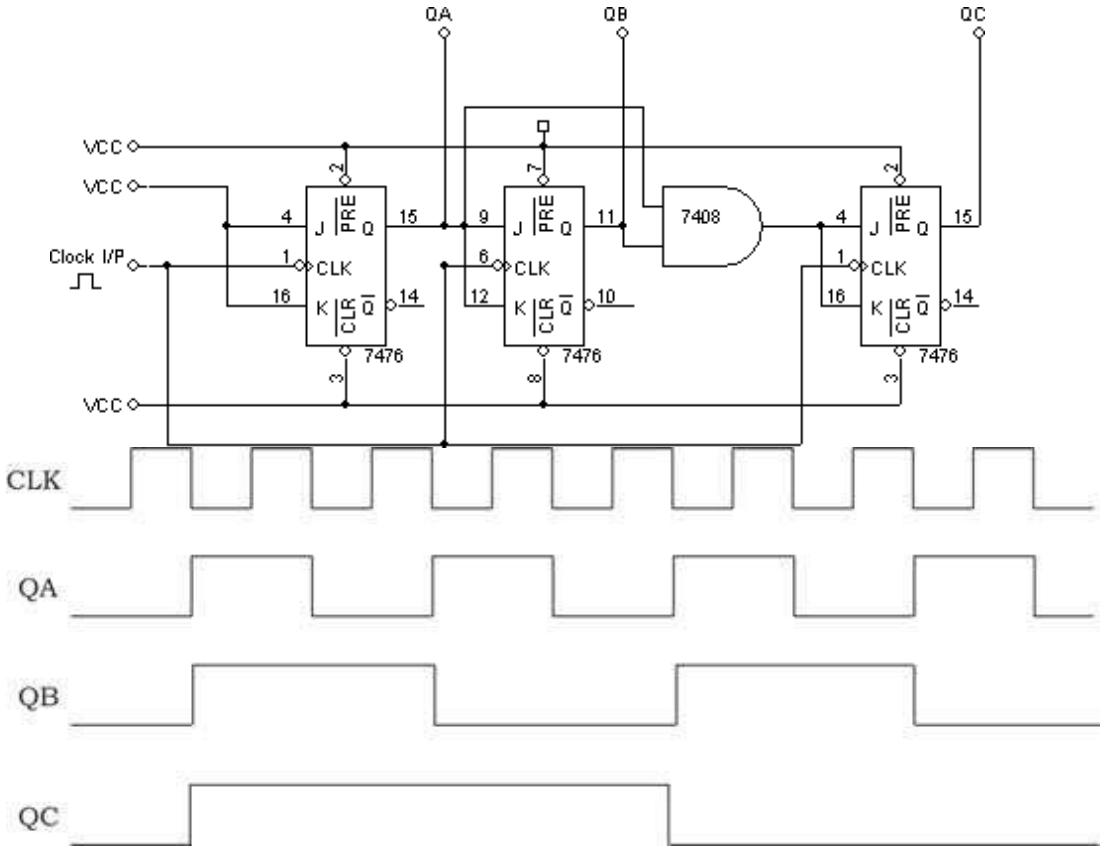
MOD 3 Asynchronous Counter:-



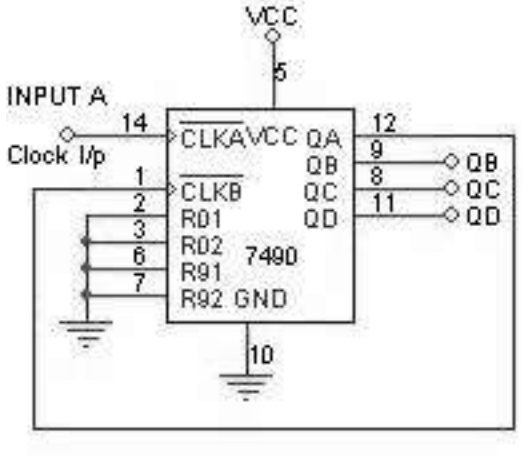
Mod 3 Asynchronous counter			
Clock	**QC**	**QB**	**QA**
0	**0**	**0**	**0**
1	**0**	**0**	**1**
2	**0**	**1**	**0**
3	**0**	**0**	**0**
4	**0**	**0**	**1**
5	**0**	**1**	**0**



3-bit Synchronous Counter:-

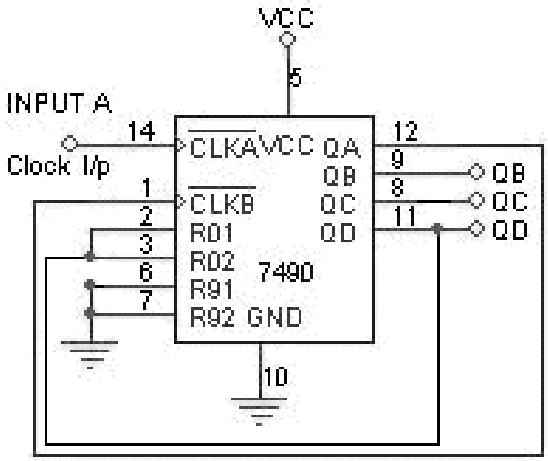


IC 7490 (Decade Counter):-



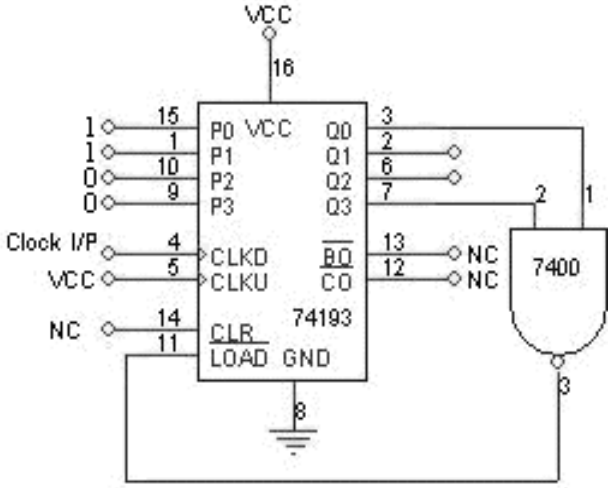
Clock	QD	QC	QB	QA
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0

IC 7490 (MOD-8 Counter):-



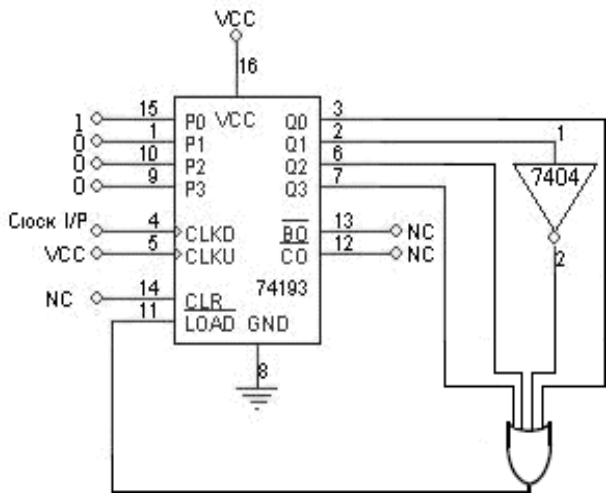
Clock	QD	QC	QB	QA
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	0	0	0	0
9	0	0	0	1

Circuit Diagram (IC 74193) To Count from 3 to 8:-



Clock	QD	QC	QB	QA	Count in Decimal
0	0	0	1	1	3
1	0	1	0	0	4
2	0	1	0	1	5
3	0	1	1	0	6
4	0	1	1	1	7
5	1	0	0	0	8
6	0	0	1	1	3
7	repeats				4

Circuit Diagram (IC 74193) To Count from 8 to 3:-

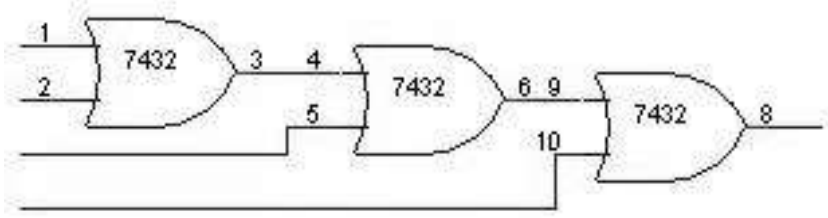


Clock	QD	QC	QB	QA	Count in Decimal
0	0	1	0	1	5
1	0	1	1	0	6
2	0	1	1	1	7
3	1	0	0	0	8
4	1	0	0	1	9
5	1	0	1	0	10
6	1	0	1	1	11
7	1	1	0	0	12
8	0	1	0	1	5
9	repeats				6

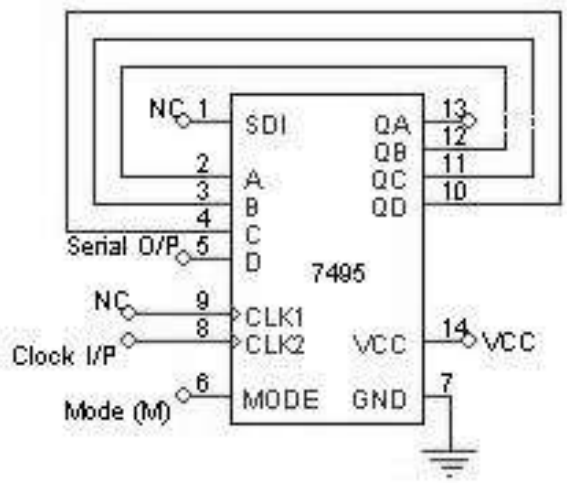
Function Table for 7490:-

Clock	R1	R2	S1	S2	QD	QC	QB	QA	
X	H	H	L	X	L	L	L	L	RESET
X	H	H	X	L	L	L	L	L	RESET
X	X	X	H	H	H	L	L	H	SET TO 9
⎍	X	L	X	L	COUNT				
⎍	L	X	L	X	COUNT				
⎍	L	X	X	L	COUNT				
⎍	X	L	L	X	COUNT				

4 I/P OR Gate can be realized as follows:-

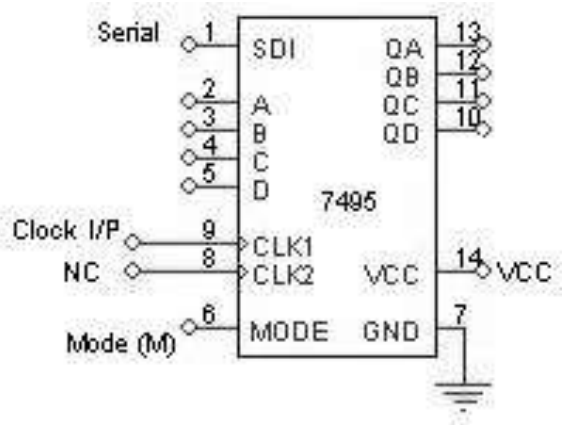


Circuit Diagram: - Shift Left



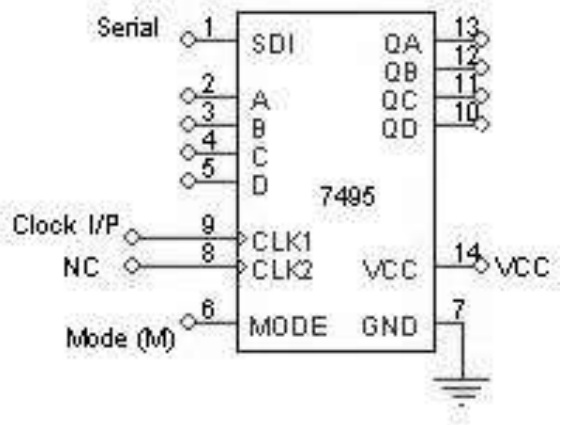
Clock	Serial i/p	QA	QB	QC	QD
1	1	X	X	X	1
2	0	X	X	1	0
3	1	X	1	0	1
4	1	1	0	1	1

SIPO (Right Shift):-



Clock	Serial i/p	QA	QB	QC	QD
1	0	0	X	X	X
2	1	1	0	X	X
3	1	1	1	0	X
4	1	1	1	1	0

SISO:-



Clock	Serial i/p	QA	QB	QC	QD
1	do=0	0	X	X	X
2	d1=1	1	0	X	X
3	d2=1	1	1	0	X
4	d3=1	1	1	1	0=do
5	X	X	1	1	1=d1
6	X	X	X	1	1=d2
7	X	X	X	X	1=d3

SHIFT REGISTERS

Aim: - Realization of 3-bit counters as a sequential circuit and Mod-N counter design (7476, 7490, 74192, 74193).

Apparatus Required: -

IC 7495, etc.

Procedure: -

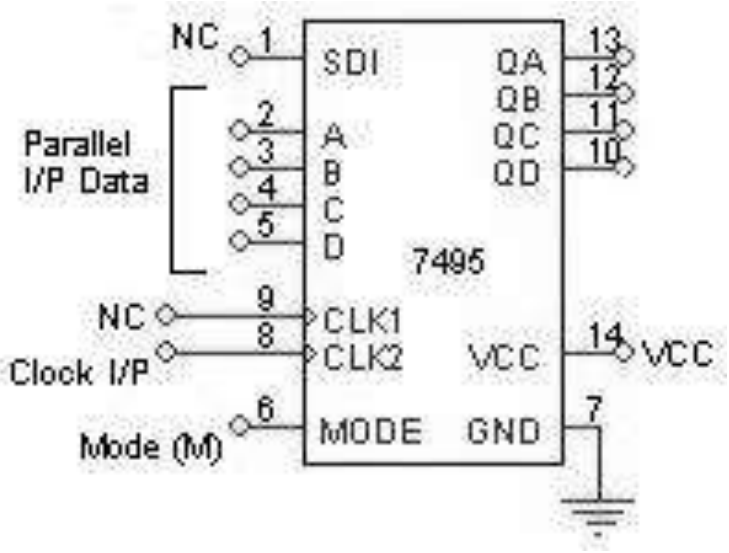
Serial In Parallel Out:-

1. **Connections are made as per circuit diagram.**
2. **Apply the data at serial i/p**
3. **Apply one clock pulse at clock 1 (Right Shift) observe this data at QA.**
4. **Apply the next data at serial i/p.**
5. **Apply one clock pulse at clock 2, observ e that the data on QA will shift to QB and the new data applied will appear at QA.**
6. **Repeat steps 2 and 3 till all the 4 bits data are entered on e by one into the shift register.**

Serial In Serial Out:-

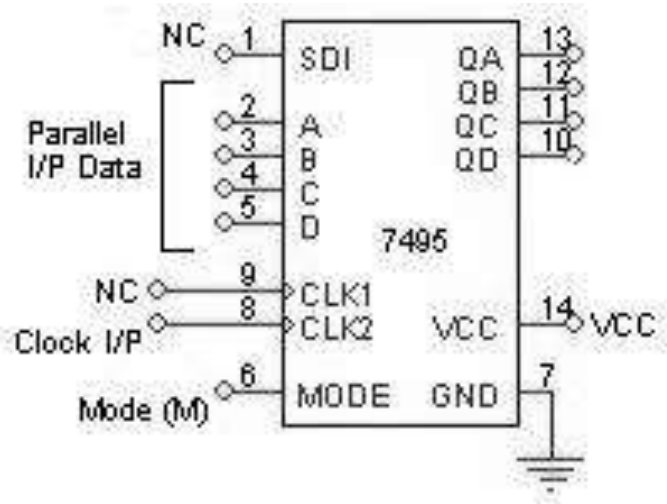
1. **Connections are made as per circuit diagram.**
2. **Load the shift register with 4 bits of data one by one serially.**
3. **At the end of 4th clock pulse the first data 'd0' appears at QD.**
4. **Apply another clock pulse; the second data 'd1' appears at QD.**
5. **Apply another clock pulse; the third data appears at QD.**
6. **Application of next clock pulse will enable the 4 th data 'd3' to appear at QD. Thus the data applied serially at the input comes out serially at QD**

PISO:-



Mode	Clock	Parallel i/p				Parallel o/p			
		A	B	C	D	QA	QB	QC	QD
1	1	1	0	1	1	1	0	1	1
0	2	X	X	X	X	X	1	0	1
0	3	X	X	X	X	X	X	1	0
0	4	X	X	X	X	X	X	X	1

PIPO:-



Clock	Parallel i/p				Parallel o/p			
	A	B	C	D	QA	QB	QC	QD
1	1	0	1	1	1	0	1	1

Parallel In Parallel Out:-

1. **Connections are made as per circuit diagram.**
2. **Apply the 4 bit data at A, B, C and D.**
3. **Apply one clock pulse at Clock 2 (Note: Mode control M=1).**
4. **The 4 bit data at A, B, C and D appears at QA, QB, QC and QD respectively.**

Parallel In Serial Out:-

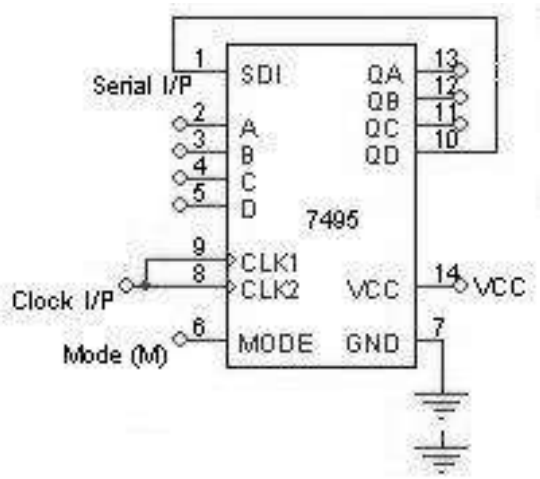
1. **Connections are made as per circuit diagram.**
2. **Apply the desired 4 bit data at A, B, C and D.**
3. **Keeping the mode control M=1 apply one clock pulse. The data applied at A, B, C and D will appear at QA, QB, QC and QD respectively.**
4. **Now mode control M=0. Apply clock pulses one by one and observe the data coming out serially at QD.**

Left Shift:-

1. **Connections are made as per circuit diagram.**
2. **Apply the first data at D and apply one clock pulse. This data appears at QD.**
3. **Now the second data is made available at D and one clock pulse applied. The data appears at QD to QC and the new data appears at QD.**
4. **Step 3 is repeated until all the 4 bits are entered one by one.**
5. **At the end 4th clock pulse, the 4 bits are available at QA, QB, QC and QD.**

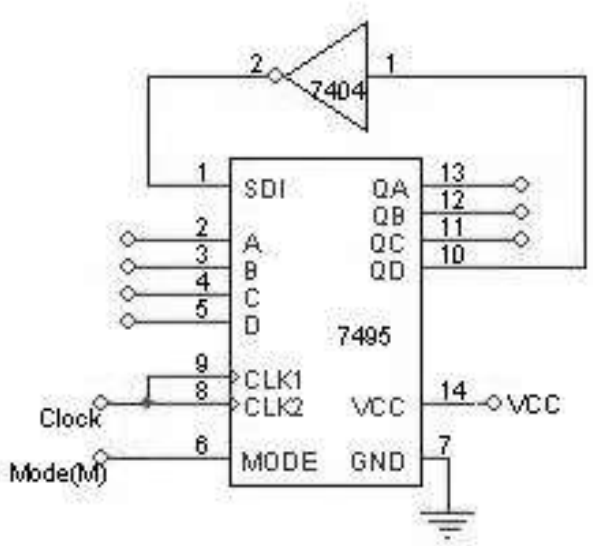
Conclusion: -

Circuit Diagram: - Ring Counter



Mode	Clock	QA	QB	QC	QD
1	1	1	0	0	0
0	2	0	1	0	0
0	3	0	0	1	0
0	4	0	0	0	1
0	5	1	0	0	0
0	6	repeats			

Johnson Counter:-



Mode	Clock	QA	QB	QC	QD
1	1	1	0	0	0
0	2	1	1	0	0
0	3	1	1	1	0
0	4	1	1	1	1
0	5	0	1	1	1
0	6	0	0	1	1
0	7	0	0	0	1
0	8	0	0	0	0
0	9	1	0	0	0
0	10	repeats			

Experiment No: Date:__/__/

JOHNSON COUNTERS / RING COUNTER

Aim:- Design and testing of Ring counter/ Johnson counter.

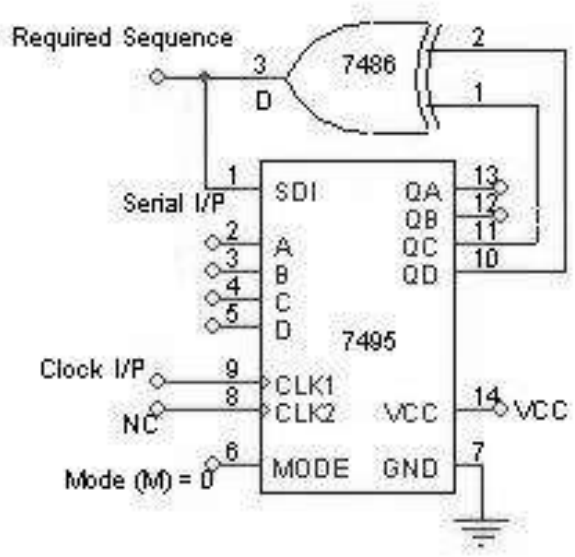
Apparatus Required: -

IC 7495, IC 7404, etc.

Procedure: -

1. **Connections are made as per the circuit diagram.**
2. **Apply the data 1000 at A, B, C and D respectively.**
3. **Keeping the mode M = 1, apply one clock pulse.**
4. **Now the mode M is made 0 and clock pulses are applied one by one, and the truth table is verified.**
5. **Above procedure is repeated for Johnson counter also.**

Circuit Diagram: - Sequence Generator



Truth Table:-

Map Value	Clock	QA	QB	QC	QD	o/p D
15	1	1	1	1	1	0
7	2	0	1	1	1	0
3	3	0	0	1	1	0
1	4	0	0	0	1	1
8	5	1	0	0	0	0
4	6	0	1	0	0	0
2	7	0	0	1	0	1
9	8	1	0	0	1	1
12	9	1	1	0	0	0
6	10	0	1	1	0	1
11	11	1	0	1	1	0
5	12	0	1	0	1	1
10	13	1	0	1	0	1
13	14	1	1	0	1	1
14	15	1	1	1	0	1

Karnaugh Map for D:-

QCQD \ QA QB	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

Experiment No: Date: __/__/

SEQUENCE GENERATOR

Aim:- Design of Sequence Generator.

Apparatus Required: -

IC 7495, IC 7486, etc.

Design:-

To generate a sequence of length S it is necessary to use at least N number of Flip-Flops, which satisfies the condition $S \leq 2^N - 1$.

The given sequence length S = 15.

Therefore N = 4.

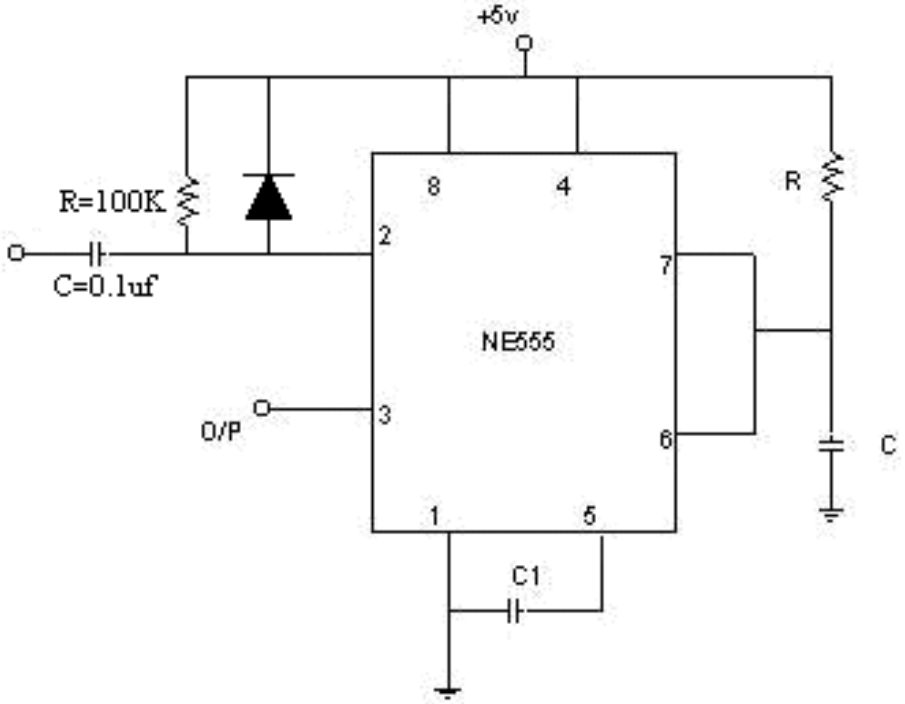
Note: - There is no guarantee that the given sequence can be generated by 4 f/fs. If the sequence is not realizable by 4 f/fs then 5 f/fs must be used and so on.

Procedure: -

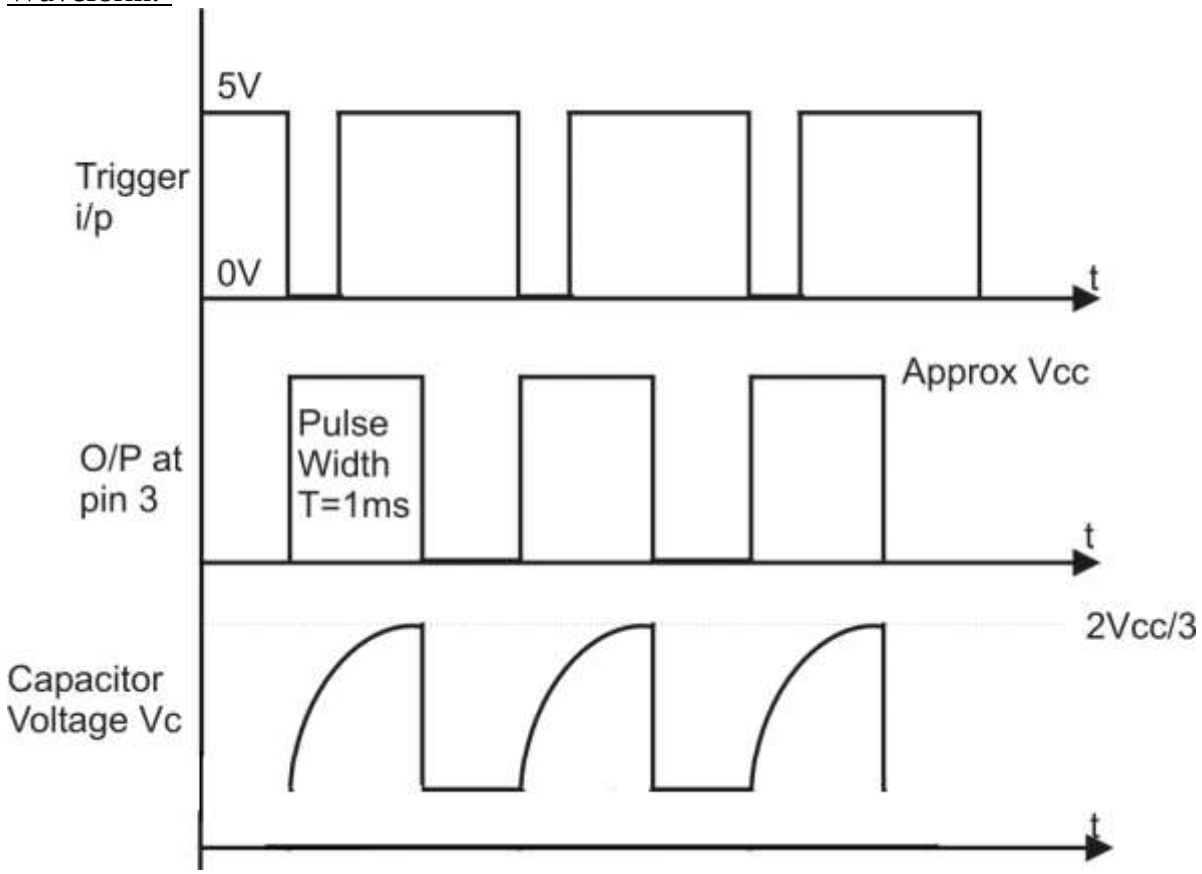
1. **Connections are made as per the circuit diagram.**
2. **Clock pulses are applied one by one and truth table is verified.**

Conclusion:-

Circuit Diagram: - Monostable Multivibrator



Waveform:-



DATA STRUCTURE LABORATORY MANUAL
(Course Code: BCACC4P)

SL. No.	Experiments
1.	Write a program to search an element from a list. Give user the option to perform Linear or Binary search. Use Template functions.
2.	WAP using templates to sort a list of elements. Give user the option to perform sorting using Insertion sort, Bubble sort or Selection sort.
3.	Implement Linked List include functions for insertion, deletion and search of a number, reverse the list and concatenate two linked lists.
4.	Implement Circular Linked List include functions for insertion and deletion.
5.	Perform Stack operations using Linked List implementation.
6.	Perform Stack operations using Array implementation.
7.	Perform Queues operations using Circular Array implementation.
8.	WAP to scan a polynomial using linked list and add two polynomials.
9.	WAP to calculate factorial of a given no. (i) using recursion, (ii) using iteration.
10.	WAP to display Fibonacci series (i)using recursion, (ii) using iteration.
11.	WAP to calculate GCD of 2 number (i) with recursion (ii) without recursion.
12.	WAP to create a Binary Search Tree and include following operations in tree: (a) Insertion (Recursive and Iterative Implementation) (b) Deletion.
13.	WAP to create Binary Tree and display its pre-order, post-order and in-order traversals Recursively.
14.	WAP to create Binary Tree and display its pre-order, post-order and in-order traversals Iteratively.
15.	WAP to implement Diagonal Matrix using one-dimensional array.
16.	WAP to implement Lower Triangular Matrix using one-dimensional array.
17.	WAP to implement Upper Triangular Matrix using one-dimensional array.

1. ***Write a program to search an element from a list. Give user the option to perform Linear or Binary search. Use Template functions.***

Program:

```
#include<iostream>
#include<conio.h>
using namespace std;
//Linear Search Method using Template class
template <class T>
 T Linear_Search(T Iarr[],T key,int n){
    for(int i=0;i<n;i++){
       if(Iarr[i]==key){
          return i;
       }
    }
    return -1;
 }
 //Binary Search Method using Tempalte class
template <class T>
T Binary_Search(T *Iarr,T key,int n)
{
 int l,mid,h;
 l=0;
 h=n-1;
 while(l<=h)
 {
 mid=(l+h)/2;
 if(key==Iarr[mid])
    return mid;
 else if(key<Iarr[mid])
   h=mid-1;
 else
   l=mid+1;
 }
   return -1;
}
int main()
{
//Integer elements
 int Iarr[20],Ielement,i,no;
```

```
cout<<"Enter the number of elements of Integer array: "<<endl;
cin>>no;
for(i=0;i<no;i++){
   cin>>Iarr[i];
}
cout<<"Elements of Integer Array "<<endl;
for(i=0;i<no;i++)
{
 cout<<Iarr[i]<<" ";
}
cout<<"\nEnter an item to be search using Linear_Search Method: ";
cin>>Ielement;
//Linear Search technique is used for Searching the Integer element
int result=Linear_Search(Iarr,Ielement,no);
if(result==-1){
   cout<<"Linear_Search Method Element not found in the list ";
}
else{
   cout<<"Linear_Search Method Element is found at position:
"<<result<<endl;
}
//Binary Search technique is used for Searching the Integer element
cout<<"\nEnter an item to be search using Binary_Search Method: ";
 cin>>Ielement;
result=Binary_Search(Iarr,Ielement,no);
if(result==-1){
   cout<<"Binary_Search Method Element not found in the list "<<endl;
}
else{
   cout<<"Binary_Search Method Element is found at position:
"<<result<<endl;
}
//Float elements
float  F_arr[10],F_element;
cout<<"Enter the "<<no<<" elements of Float array: "<<endl;
for(i=0;i<no;i++){
   cin>>F_arr[i];
}
cout<<"\nElements of Float Array \n";
for(int i=0;i<no;i++)
```

```
 {
  cout<<F_arr[i]<<" ";
 }
 //Linear Search technique is used for Searching the Floating element
 cout<<"\nEnter an item to be search using Linear_Search Method: ";
 cin>>F_element;
 result=Linear_Search(F_arr,F_element,no);
 if(result==-1){
    cout<<"Linear_Search Method Element not found in the list ";
 }
 else{
    cout<<"Linear_Search Method Element is found at position:
"<<result;
 }
 //Binary Search technique is used for Searching the Floating element
 cout<<"\nEnter an item to be search using Binary_Search Method: ";
 cin>>F_element;
result=Binary_Search(F_arr,F_element,no);
if(result==-1){
    cout<<"Binary_Search Method Element not found in the list "<<endl;
 }
 else{
    cout<<"Binary_Search Method Element is found at position:
"<<result<<endl;
 }
//Character elements
char Carr[10]={'a','b','c','d','e','f','g','h','i','j'};
char Celement;
cout<<"Elements of Character Array "<<endl;
 for(i=0;i<10;i++)
 {
  cout<<Carr[i]<<" ";
 }
 //Linear Search technique is used for Searching the Character element
 cout<<"\nEnter an item to be search using Linear_Search Method: ";
 cin>>Celement;
 result=Linear_Search(Carr,Celement,10);
 if(result==-1){
    cout<<"Linear_Search Method Element not found in the list ";
 }
```

```
 else{
    cout<<"Linear_Search Method Element is found at position:
"<<result<<endl;
 }
 //Binary search technique is used for Searching the Character element
 cout<<"\nEnter an item to be search using Binary_Search Method: ";
 cin>>Celement;
result=Binary_Search(Carr,Celement,10);
if(result==-1){
    cout<<"Binary_Search Method Element not found in the list "<<endl;
 }
 else{
    cout<<"Binary_Search Method Element is found at position:
"<<result<<endl;
 }
 getch();
 return 0;
}
```

Input and Output Section:

```
Enter the number of elements of Integer array:
8
10 20 30 40 50 60 70 80
Elements of Integer Array
10 20 30 40 50 60 70 80
Enter an item to be search using Linear_Search Method: 20
Linear_Search Method Element is found at position: 1
Enter an item to be search using Binary_Search Method: 70
Binary_Search Method Element is found at position: 6
Enter the 8 elements of Float array:
1.1 2.3 4.5 6.4 6.55 7.1 7.11 2.1
Elements of Float Array
1.1 2.3 4.5 6.4 6.55 7.1 7.11 2.1
Enter an item to be search using Linear_Search Method: 2.1
Linear_Search Method Element is found at position: 7
Enter an item to be search using Binary_Search Method: 3.7
Binary_Search Method Element not found in the list
Elements of Character Array
a b c d e f g h i j
Enter an item to be search using Linear_Search Method: f
```

```
Linear_Search Method Element is found at position: 5
Enter an item to be search using Binary_Search Method: k
Binary_Search Method Element not found in the list
```

2. ***WAP using templates to sort a list of elements. Give user the option to perform sorting using Insertion sort, Bubble sort or Selection sort.***

Program:

```
#include <iostream>
#include<conio.h>
using namespace std;
template <class T>
void swap(T *x,T *y)
{
   T temp=*x;
   *x=*y;
   *y=temp;
}
//Bubble sort Method using template class
template <class T>
void Bubble(T A[],int n)
{
 int i,j;
 for(i=0;i<n-1;i++)
 {
   for(j=0;j<n-i-1;j++)
   {
      if(A[j]>A[j+1]){
          swap(&A[j],&A[j+1]);
         }
    }
 }
}
//Insersion sort Method using Template class
template <class T>
void Insersion(T A[],int n)
{
 int i,j;
 T x;
```

```
 for(i=1;i<n;i++)
 {
   j=i-1;
   x=A[i];
      while(j>-1 && A[j]>x)
   {
      A[j+1]=A[j];
      j--;
   }
   A[j+1]=x;
 }
}
//Selection Sort Method using Template Class
template <class T>
void SelectionSort(T A[],int n)
{
 int i,j,k;
 for(i=0;i<n-1;i++)
 {
   for(j=k=i;j<n;j++)
   {
      if(A[j]<A[k])
       k=j;
   }
   swap(&A[i],&A[k]);
 }
}

int main()
{

   //Bubble sort technique is used for Integer type array
   int A[100],i,n;
   cout<<"Enter the how many integer elements to be sort using Bubble
sorting:"<<endl;
   cin>>n;
   cout<<"Elements of integer array are: "<<endl;
   for(i=0;i<n;i++){
      cin>>A[i];
   }
```

```
    cout<<"Before Bubble sorting the elements are: "<<endl;
    for(i=0;i<n;i++){
       cout<<A[i]<<" ";
    }
    Bubble(A,n);
    cout<<"\nAfter Bubble sorting the elements are: "<<endl;
    for(i=0;i<n;i++)
    printf("%d ",A[i]);
    printf("\n");
    //Bubble sort technique is used for Floating type array
    float B[100];
    cout<<"Enter the how many float elements to be sort using Bubble
sorting:"<<endl;
    cin>>n;
    cout<<"Elements of floating array are: "<<endl;
    for(i=0;i<n;i++){
       cin>>B[i];
    }
    cout<<"Before Bubble sorting the elements are: "<<endl;
    for(i=0;i<n;i++){
       cout<<B[i]<<" ";
    }
    Bubble(B,n);
    cout<<"\nAfter Bubble sorting the elements are: "<<endl;
    for(i=0;i<n;i++)
    printf("%f ",B[i]);
    printf("\n");

//Insersion sort technique is used for integer type array
    //int A[100],i,n;
    cout<<"Enter the how many integer elements to be sort using Insersion
sorting:"<<endl;
    cin>>n;
    cout<<"Elements of integer array are: "<<endl;
    for(i=0;i<n;i++){
       cin>>A[i];
    }
    cout<<"Before Insersion sorting the elements are: "<<endl;
    for(i=0;i<n;i++){
       cout<<A[i]<<" ";
```

```
    }
    Insersion(A,n);
    cout<<"\nAfter Insersion sorting the elements are: "<<endl;
    for(i=0;i<n;i++)
    printf("%d ",A[i]);
    printf("\n");
    //Insersion sort technique is used for Floating type array
    //float B[100];
    cout<<"Enter the how many float elements to be sort using Insersion
sorting:"<<endl;
    cin>>n;
    cout<<"Elements of floating array are: "<<endl;
    for(i=0;i<n;i++){
       cin>>B[i];
    }
    cout<<"Before Insersion sorting the elements are: "<<endl;
    for(i=0;i<n;i++){
       cout<<B[i]<<" ";
    }
    Insersion(B,n);
    cout<<"\nAfter Insersion sorting the elements are: "<<endl;
    for(i=0;i<n;i++)
    printf("%f ",B[i]);
    printf("\n");

//Selection sort technique is used for integer type array
    //int A[100],i,n;
    cout<<"Enter the how many integer elements to be sort using Selection
sorting:"<<endl;
    cin>>n;
    cout<<"Elements of integer array are: "<<endl;
    for(i=0;i<n;i++){
       cin>>A[i];
    }
    cout<<"Before Selection sorting the elements are: "<<endl;
    for(i=0;i<n;i++){
       cout<<A[i]<<" ";
    }
    SelectionSort(A,n);
    cout<<"\nAfter Selection sorting the elements are: "<<endl;
```

```
    for(i=0;i<n;i++)
    printf("%d ",A[i]);
    printf("\n");
    //Insersion sort technique is used for Floating type array
   // float B[100];
    cout<<"Enter the how many float elements to be sort using Selection
sorting:"<<endl;
    cin>>n;
    cout<<"Elements of floating array are: "<<endl;
    for(i=0;i<n;i++){
       cin>>B[i];
    }
    cout<<"Before Selection sorting the elements are: "<<endl;
    for(i=0;i<n;i++){
       cout<<B[i]<<" ";
    }
    SelectionSort(B,n);
    cout<<"\nAfter Selection sorting the elements are: "<<endl;
    for(i=0;i<n;i++)
    printf("%f ",B[i]);
    printf("\n");
    getch();
    return 0;
}
```

Input and Output Section:

Enter the how many integer elements to be sort using Bubble sorting:
5
Elements of integer array are:
10 23 09 11 15
Before Bubble sorting the elements are:
10 23 9 11 15
After Bubble sorting the elements are:
9 10 11 15 23
Enter the how many float elements to be sort using Bubble sorting:
5
Elements of floating array are:
9.1 2.3 4.5 1.1 2.9
Before Bubble sorting the elements are:
9.1 2.3 4.5 1.1 2.9

After Bubble sorting the elements are:
1.100000 2.300000 2.900000 4.500000 9.100000
Enter the how many integer elements to be sort using Insersion sorting:
6
Elements of integer array are:
19 24 10 11 14
23
Before Insersion sorting the elements are:
19 24 10 11 14 23
After Insersion sorting the elements are:
10 11 14 19 23 24
Enter the how many float elements to be sort using Insersion sorting:
3
Elements of floating array are:
2.3 1.2 .9
Before Insersion sorting the elements are:
2.3 1.2 0.9
After Insersion sorting the elements are:
0.900000 1.200000 2.300000
Enter the how many integer elements to be sort using Selection sorting:
10
Elements of integer array are:
10 20 50 30 21 35 44 9 99 102
Before Selection sorting the elements are:
10 20 50 30 21 35 44 9 99 102
After Selection sorting the elements are:
9 10 20 21 30 35 44 50 99 102
Enter the how many float elements to be sort using Selection sorting:
5
Elements of floating array are:
1.1 .9 .2 .99 1.234
Before Selection sorting the elements are:
1.1 0.9 0.2 0.99 1.234
After Selection sorting the elements are:
0.200000 0.900000 0.990000 1.100000 1.234000

3. Implement Linked List include functions for insertion, deletion and search of a number, reverse the list and concatenate two linked lists.

Insertion Operation using linked list:

Program:

```
/* linked List Insertion */
#include <stdio.h>
#include <stdlib.h>
struct Node
{
 int data;
 struct Node *next;
}*first=NULL;

void create(int A[],int n)
{
 int i;
 struct Node *t,*last;
 first=(struct Node *)malloc(sizeof(struct Node));
 first->data=A[0];
 first->next=NULL;
 last=first;
 for(i=1;i<n;i++)
 {
   t=(struct Node*)malloc(sizeof(struct Node));
   t->data=A[i];
   t->next=NULL;
   last->next=t;
   last=t;
 }
}
int count(struct Node *p)
{
 int l=0;
 while(p)
 {
   l++;
   p=p->next;
 }
```

```
 return l;
}
void Display(struct Node *p)
{
 while(p!=NULL)
 {
   printf("%d ",p->data);
   p=p->next;
 }
}
void Insert(struct Node *p,int index,int x)
{
 struct Node *t;
 int i;

 if(index < 0 || index > count(p))
   return;
 t=(struct Node *)malloc(sizeof(struct Node));
 t->data=x;
 if(index == 0)
 {
   t->next=first;
   first=t;
 }
 else
 {
   for(i=0;i<index-1;i++)
   p=p->next;
   t->next=p->next;
   p->next=t;
 }
}
int main()
{

 //int A[]={10,20,30,40,50};
 //create(A,5);

//Insert Node
 printf("Node Insert: \n");
```

```
Insert(first,0,5);
Display(first);
printf("\nNode Insert: \n");
Insert(first,1,10);
Display(first);
//First Node Insert
printf("\nFirst Node Insert: \n");
Insert(first,0,15);
Display(first);
//Last Node Insert
printf("\nLast Node Insert: \n");
Insert(first,3,20);
Display(first);
//Any position Node Insert
printf("\nGiven position Node Insert: \n");
Insert(first,2,39);
Display(first);
return 0;
}
```

Input and Output Section:

Node Insert:
5
Node Insert:
5 10
First Node Insert:
15 5 10
Last Node Insert:
15 5 10 20
Given position Node Insert:
15 5 39 10 20

Deletion Operation using linked list

Program:

```
/* Linked List Delete element */
#include <stdio.h>
#include <stdlib.h>
struct Node
{
 int data;
 struct Node *next;
}*first=NULL;
void create(int A[],int n)
{
 int i;
 struct Node *t,*last;
 first=(struct Node *)malloc(sizeof(struct Node));
 first->data=A[0];
 first->next=NULL;
 last=first;
 for(i=1;i<n;i++)
 {
    t=(struct Node*)malloc(sizeof(struct Node));
    t->data=A[i];
    t->next=NULL;
    last->next=t;
    last=t;
 }
}
void Display(struct Node *p)
{
 while(p!=NULL)
 {
    printf("%d ",p->data);
    p=p->next;
 }
}
int count(struct Node *p)
{
 int l=0;
 while(p)
```

```
 {
   l++;
   p=p->next;
 }
 return l;
}
int Delete(struct Node *p,int index)
{
 struct Node *q=NULL;
 int x=-1,i;
 if(index < 1 || index > count(p))
   return -1;
 if(index==1)
 {
   q=first;
   x=first->data;
   first=first->next;
   free(q);
   return x;
 }
 else
 {
   for(i=0;i<index-1;i++)
   {
   q=p;
   p=p->next;
   }
   q->next=p->next;
   x=p->data;
   free(p);
   return x;
 }
}
int main()
{
 int i;
 int A[]={10,20,30,40,50};
 create(A,5);
 printf("Lined List element are: \n");
 Display(first);
```

```
 printf("\nDelete element %d\n",Delete(first,2));
 Display(first);
 printf("\nDelete element %d\n",Delete(first,1));
 Display(first);
 printf("\nDelete element %d\n",Delete(first,0));
 Display(first);
return 0;
}
```

Input and Output Section:

Lined List element are:
10 20 30 40 50
Delete element 20
10 30 40 50
Delete element 10
30 40 50
Delete element -1
30 40 50

Search of a number using linked list

Program:

```
/*Searching of a number using Lined List */
#include <stdio.h>
#include <stdlib.h>
struct Node
{
 int data;
 struct Node *next;
}*first=NULL;
void create(int A[],int n)
{
 int i;
 struct Node *t,*last;
 first=(struct Node *)malloc(sizeof(struct Node));
 first->data=A[0];
 first->next=NULL;
 last=first;
 for(i=1;i<n;i++)
 {
```

```
    t=(struct Node*)malloc(sizeof(struct Node));
    t->data=A[i];
    t->next=NULL;
    last->next=t;
    last=t;
  }
}
struct Node * LSearch(struct Node *p,int key)
{
 while(p!=NULL)
 {
     if(key==p->data){
           return p;
     }
     else{
        p=p->next;
     }
 }
    return NULL;
}
struct Node * RSearch(struct Node *p,int key)
{
 if(p==NULL)
    return NULL;
 if(key==p->data)
    return p;
 return RSearch(p->next,key);

}
int main()
{
 struct Node *temp;
 int A[]={3,5,7,10,25,8,32,2},i;
 printf("Linked list elements are: \n");
 for(i=0;i<8;i++){
     printf(" %d ",A[i]);
 }
 //Recursive LinearSearch
 create(A,8);
```

```
 temp=RSearch(first,8);
 if(temp)
    printf("\nKey is Found* %d \n",temp->data);
else
    printf("Key is not Found: \n");
//Iterative LinearSearch
temp=LSearch(first,27);
if(temp)
    printf("\nKey is Found* %d \n",temp->data);
else
    printf("Key is not found: ");

 return 0;
}
```

Input and Output Section:

```
Linked list elements are:
 3  5  7  10  25  8  32  2
Key is Found* 8
Key is not found:
```

Reverse a linked list:

Program:

```
/* Reverse a Linked List */
#include <stdio.h>
#include <stdlib.h>
struct Node
{
 int data;
 struct Node *next;
}*first=NULL,*second=NULL,*third=NULL;
void Display(struct Node *p)
{
 while(p!=NULL)
 {
 printf("%d ",p->data);
 p=p->next;
 }
}
```

```
void create(int A[],int n)
{
 int i;
 struct Node *t,*last;
 first=(struct Node *)malloc(sizeof(struct Node));
 first->data=A[0];
 first->next=NULL;
 last=first;

 for(i=1;i<n;i++)
 {
 t=(struct Node*)malloc(sizeof(struct Node));
 t->data=A[i];
 t->next=NULL;
 last->next=t;
 last=t;
 }
}
int count(struct Node *p)
{
 int l=0;
 while(p)
 {
   l++;
   p=p->next;
 }
 return l;
}
void Reverse1(struct Node *p)
{
 int *A,i=0;
 struct Node *q=p;

 A=(int *)malloc(sizeof(int)*count(p));

 while(q!=NULL)
 {
 A[i]=q->data;
 q=q->next;
 i++;
```

```
 }
 q=p;
 i--;
 while(q!=NULL)
 {
 q->data=A[i];
 q=q->next;
 i--;
 }
}
void Reverse2(struct Node *p)
{
 struct Node *q=NULL,*r=NULL;

 while(p!=NULL)
 {
 r=q;
 q=p;
 p=p->next;
 q->next=r;
 }
 first=q;
}
void Reverse3(struct Node *q,struct Node *p)
{
 if(p)
 {
 Reverse3(p,p->next);
 p->next=q;
 }
 else
 first=q;
}

int main()
{

 int A[]={10,20,40,50,60};
 create(A,5);
 printf("Linked list elements are: \n");
```

```
 Display(first);

 Reverse3(NULL,first);
 printf("\nReverse Linked list elements are: \n");
 Display(first);

 return 0;
}
```

Input and Output Section:

Linked list elements are:
10 20 40 50 60
Reverse Linked list elements are:
60 50 40 20 10

Concatenate of two linked list:

Program:

```
/*Concatenate two Linked List*/
#include <stdio.h>
#include <stdlib.h>
struct Node
{
 int data;
 struct Node *next;
}*first=NULL,*second=NULL,*third=NULL;
void Display(struct Node *p)
{
 while(p!=NULL)
 {
 printf("%d ",p->data);
 p=p->next;
 }
}
void create(int A[],int n)
{
 int i;
 struct Node *t,*last;
 first=(struct Node *)malloc(sizeof(struct Node));
 first->data=A[0];
```

```
 first->next=NULL;
 last=first;

 for(i=1;i<n;i++)
 {
 t=(struct Node*)malloc(sizeof(struct Node));
 t->data=A[i];
 t->next=NULL;
 last->next=t;
 last=t;
 }
}
void create2(int A[],int n)
{
 int i;
 struct Node *t,*last;
 second=(struct Node *)malloc(sizeof(struct Node));
 second->data=A[0];
 second->next=NULL;
 last=second;

 for(i=1;i<n;i++)
 {
 t=(struct Node*)malloc(sizeof(struct Node));
 t->data=A[i];
 t->next=NULL;
 last->next=t;
 last=t;
 }
}
/*void Merge(struct Node *p,struct Node *q)
{
 struct Node *last;
 if(p->data < q->data)
 {
 third=last=p;
 p=p->next;
 third->next=NULL;
 }
 else
```

```
{
third=last=q;
q=q->next;
third->next=NULL;
}
while(p && q)
{
if(p->data < q->data)
{
last->next=p;
last=p;
p=p->next;
last->next=NULL;

}
else
{
last->next=q;
last=q;
q=q->next;
last->next=NULL;

}
}
if(p)last->next=p;
if(q)last->next=q;

}
*/
void Concat(struct Node *p,struct Node *q){
   third=p;
   while(p->next!=NULL){
      p=p->next;
   }
   p->next=q;

}
int main()
{
```

```
int A[]={10,20,40,50,60};
int B[]={15,18,25,30,55};
create(A,5);
create2(B,5);

//Merge(frist,second);
//Display(third);
printf("First Linked List \n");
Display(first);
printf("\n");
printf("Second Linked List \n");
Display(second);
printf("\n");
Concat(first,second);
printf("Concatenate two Linked List: \n");
Display(third);
printf("\n\n");
return 0;
}
```

Input and Output Section:

First Linked List
10 20 40 50 60
Second Linked List
15 18 25 30 55
Concatenate two Linked List:
10 20 40 50 60 15 18 25 30 55

4. ***Implement Circular Linked List include functions for insertion and deletion.***

Circular Linked List for Insertion Operation:

Program:

```
/*Circular Linked List*/
#include <stdio.h>
#include <stdlib.h>
#include<conio.h>
struct Node
{
```

```
 int data;
 struct Node *next;
}*Head;
void create(int A[],int n)
{
 int i;
 struct Node *t,*last;
 Head=(struct Node*)malloc(sizeof(struct Node));
 Head->data=A[0];
 Head->next=Head;
 last=Head;

 for(i=1;i<n;i++)
 {
   t=(struct Node*)malloc(sizeof(struct Node));
   t->data=A[i];
   t->next=last->next;
   last->next=t;
   last=t;
 }
}
void Display(struct Node *h)
{
 do
 {
   printf("%d ",h->data);
   h=h->next;
 }while(h!=Head);
   printf("\n");
}
/*void RDisplay(struct Node *h)
{
 static int flag=0;
 if(h!=Head || flag==0)
 {
 flag=1;
 printf("%d ",h->data);
 RDisplay(h->next);
 }
 flag=0;
```

```
}
*/
int Length(struct Node *p)
{
   int len=0;
 do
 {
   len++;
   p=p->next;

 }while(p!=Head);
   return len;
}
void Insert(struct Node *p,int index, int x)
{
 struct Node *t;
 int i;
 if(index<0 || index > Length(p))
 return;

 if(index==0)
 {
   t=(struct Node *)malloc(sizeof(struct Node));
   t->data=x;
   if(Head==NULL)
   {
      Head=t;
      Head->next=Head;
   }
 else
 {
   while(p->next!=Head)p=p->next;
   p->next=t;
   t->next=Head;
   Head=t;
 }
 }
 else
 {
   for(i=0;i<index-1;i++)
```

```
   p=p->next;
    t=(struct Node *)malloc(sizeof(struct Node));
    t->data=x;
    t->next=p->next;
    p->next=t;
 }
}

int main()
{
 int A[]={2,3,4,5,6};
 create(A,5);
 printf("Linked List are: \n");
 Display(Head);

//Insert Node
 printf("\nNode Insert: \n");
 Insert(Head,2,10);
 Display(Head);
 //First Node Insert
 printf("\nFirst Node Insert: \n");
 Insert(Head,0,15);
 Display(Head);
 //Last Node Insert
 printf("\nLast Node Insert: \n");
 Insert(Head,7,20);
 Display(Head);
 //Any position Node Insert
 printf("\nGiven position Node Insert: \n");
 Insert(Head,2,39);
 Display(Head);
getch();
return 0;
}
```

Input and Output Section:
Linked List are:
2 3 4 5 6

Node Insert:
2 3 10 4 5 6

First Node Insert:
15 2 3 10 4 5 6

Last Node Insert:
15 2 3 10 4 5 6 20

Given position Node Insert:
15 2 39 3 10 4 5 6 20

Circular Linked List for Deletion Operation:

Program:

```
/*Circular Linked List Deletion*/
#include <stdio.h>
#include <stdlib.h>
#include<conio.h>
struct Node
{
 int data;
 struct Node *next;
}*Head;
void create(int A[],int n)
{
 int i;
 struct Node *t,*last;
 Head=(struct Node*)malloc(sizeof(struct Node));
 Head->data=A[0];
 Head->next=Head;
 last=Head;
 for(i=1;i<n;i++)
 {
   t=(struct Node*)malloc(sizeof(struct Node));
```

```
    t->data=A[i];
    t->next=last->next;
    last->next=t;
    last=t;
  }
}
void Display(struct Node *h)
{
  do
  {
    printf("%d ",h->data);
    h=h->next;
  }while(h!=Head);
    printf("\n");
}
int Length(struct Node *p)
{
    int len=0;
  do
  {
    len++;
    p=p->next;

  }while(p!=Head);
    return len;
}

int Delete(struct Node *p,int index)
{
    struct Node *q;
    int i,x;

  if(index <0 || index >Length(Head))
    return -1;
  if(index==1)
  {
    while(p->next!=Head)p=p->next;
    x=Head->data;
  if(Head==p)
  {
```

```
    free(Head);
    Head=NULL;
  }
  else
  {
    p->next=Head->next;
    free(Head);
    Head=p->next;
    }
  }
  else
  {
    for(i=0;i<index-2;i++)
    p=p->next;
    q=p->next;
    p->next=q->next;
    x=q->data;
    free(q);
  }
  return x;
}
int main()
{
  int A[]={2,3,4,5,6,7,8,9,10};
  create(A,9);
  printf("Linked List are: \n");
  Display(Head);

  //First Node Delete
  printf("\nFirst Node Delete: \n");
  Delete(Head,1);
  Display(Head);

  //Last Node Delete
  printf("\nLast Node Delete: \n");
  Delete(Head,8);
  Display(Head);

  //Any position Node Insert
  printf("\nGiven position Node Delete: \n");
```

```
 Delete(Head,5);
 Display(Head);
 getch();
return 0;
}
```

Input and Output Section:

Linked List are:
2 3 4 5 6 7 8 9 10

First Node Delete:
3 4 5 6 7 8 9 10

Last Node Delete:
3 4 5 6 7 8 9

Given position Node Delete:
3 4 5 6 8 9

5. Perform Stack operations using Linked List implementation.

Program:

```
/*Stack using Linked List*/
#include <stdio.h>
#include <stdlib.h>
struct Node
{
 int data;
 struct Node *next;
}*top=NULL;

void push(int x)
{
 struct Node *t;
 t=(struct Node*)malloc(sizeof(struct Node));
 if(t==NULL)
   printf("stack is full\n");
 else
 {
   t->data=x;
```

```
    t->next=top;
    top=t;
  }
}

int pop()
{
 struct Node *t;
 int x=-1;
 if(top==NULL)
    printf("Stack is Empty\n");
 else
 {
    t=top;
    top=top->next;
    x=t->data;
    free(t);
 }
 return x;
}

void Display()
{
 struct Node *p;
 p=top;
 while(p!=NULL)
 {
    printf("%d ",p->data);
    p=p->next;
 }
 printf("\n");
}
int main()
{
// push the elements
printf("Push the elements are: \n");
//last elements top of the stack
 push(10);
 push(20);
 push(30);
```

```
 push(40);
 push(50);
 Display();

 printf("pop element %d \n",pop());
 printf("pop element %d %d ",pop(),pop()); //right to left operation
 return 0;
}
```

Input and Output Section:

Push the elements are:
50 40 30 20 10
pop element 50
pop element 30 40

6. Perform Stack operations using Array implementation.

Program:

```
/* Stack using Array */
#include <stdio.h>
#include <stdlib.h>
struct Stack
{
 int size;
 int top;
 int *S;
};
void create(struct Stack *st)
{
 printf("Enter Size ");
 scanf("%d",&st->size);
 st->top=-1;
 st->S=(int *)malloc(st->size*sizeof(int));
}
void Display(struct Stack st)
{
 int i;
 for(i=st.top;i>=0;i--)
    printf("%d ",st.S[i]);
 printf("\n");
```

```
}
void push(struct Stack *st,int x)
{
 if(st->top==st->size-1)
   printf("Stack overflow\n");
 else
 {
   st->top++;
   st->S[st->top]=x;
 }
}
int pop(struct Stack *st)
{
   int x=-1;
   if(st->top==-1)
      printf("Stack Underflow\n");
   else
   {
      x=st->S[st->top--];
   }
 return x;
}

int peek(struct Stack st,int index)
{
 int x=-1;
 if(st.top-index+1<0)
   printf("Invalid Index \n");
   x=st.S[st.top-index+1];
 return x;
}
int isEmpty(struct Stack st)
{
 if(st.top==-1)
   return 1;
 return 0;
}
int isFull(struct Stack st)
{
```

```
    return st.top==st.size-1;
}
int stackTop(struct Stack st)
{
    if(!isEmpty(st))
        return st.S[st.top];
 return -1;
}
int main()
{
 struct Stack st;
 create(&st);
 //push the elements into the Stack
 printf("Stack elements are: \n");
 push(&st,10);
 push(&st,20);
 push(&st,30);
 push(&st,40);
 push(&st,50);
 push(&st,60);
 //Stack overflow when Enter the size 5
 Display(st);
 //pop the elements from a Stack
 printf("pop element: %d \n",pop(&st));
 printf("After the pop operation Stack elements are: \n");
 Display(st);
 //Stack Underflow when more then 6 elements pop out
 //printf("pop element: %d %d %d %d %d
%d\n",pop(&st),pop(&st),pop(&st),pop(&st),pop(&st),pop(&st));

 //Peek the element into the Stack
 printf("peek element: %d \n",peek(st,1));
 printf("After the peek operation Stack elements are: \n");
 Display(st);
 return 0;
}
```

Input and Output Section:

```
Enter Size 6
Stack elements are:
60 50 40 30 20 10
pop element: 60
After the pop operation Stack elements are:
50 40 30 20 10
peek element: 50
After the peek operation Stack elements are:
50 40 30 20 10

Enter Size 5
Stack elements are:
Stack overflow
50 40 30 20 10
pop element: 50
After the pop operation Stack elements are:
40 30 20 10
peek element: 40
After the peek operation Stack elements are:
40 30 20 10

Enter Size 6
Stack elements are:
60 50 40 30 20 10
pop element: 60
After the pop operation Stack elements are:
50 40 30 20 10
Stack Underflow
pop element: -1 10 20 30 40 50
Invalid Index
peek element: 0
After the peek operation Stack elements are:
```

7. Perform Queues operations using Circular Array implementation.

Program:

```
/* Queue Operation Using Circular Array Implementation*/
#include <stdio.h>
#include <stdlib.h>
struct Queue
{
 int size;
 int front;
 int rear;
 int *Q;
};
void create(struct Queue *q,int size)
{
 q->size=size;
 q->front=q->rear=0;
 q->Q=(int *)malloc(q->size*sizeof(int));
}
void enqueue(struct Queue *q,int x)
{
 if((q->rear+1)%q->size==q->front)
 printf("Queue is Full\n");
 else
 {
 q->rear=(q->rear+1)%q->size;
 q->Q[q->rear]=x;
 }
}
int dequeue(struct Queue *q)
{
 int x=-1;
 if(q->front==q->rear)
 printf("Queue is Empty\n");
 else
 {
 q->front=(q->front+1)%q->size;
 x=q->Q[q->front];
 }
```

```
 return x;
}
void Display(struct Queue q)
{
 int i=q.front+1;

 do
 {
 printf("%d ",q.Q[i]);
 i=(i+1)%q.size;
 }while(i!=(q.rear+1)%q.size);

 printf("\n");
}
int main()
{
 struct Queue q;
 create(&q,6);
 printf("\nQueue elements are: \n");
 //enqueue Operation
 enqueue(&q,10);
 enqueue(&q,20);
 enqueue(&q,30);
 enqueue(&q,40);
 enqueue(&q,50);
 //enqueue(&q,60);
 Display(q);
 //dequeue Operation
 printf("dequeue element: %d \n",dequeue(&q));
 printf("After dequeue Queue elements are: \n");
 Display(q);
 return 0;
}
```

Input and Output Section:

Queue elements are:
10 20 30 40 50
dequeue element: 10
After dequeue Queue elements are:
20 30 40 50

8. WAP to scan a polynomial using linked list and add two polynomials.

Program:

```
//Polynomial addition using linked list
# include <stdio.h>
# include <stdlib.h>
struct node
{
        float coef;
        int expo;
        struct node *link;
};

 int display(struct node *ptr)
{
        if(ptr==NULL)
        {
                printf("Empty\n");
                return 0;
        }
        while(ptr!=NULL)
        {
                printf("(%.1fx^%d) + ", ptr->coef,ptr->expo);
                ptr=ptr->link;
        }
        printf("\b\b \n"); // \b\b to erase the last + sign
}
struct node *insert(struct node *start,float co,int ex)
{
        struct node *ptr,*tmp;
        tmp= (struct node*)malloc(sizeof(struct node));
        tmp->coef=co;
        tmp->expo=ex;

        //list empty or exp greater than first one
        if(start==NULL || ex>start->expo)
        {
                tmp->link=start;
                start=tmp;
        }
```

```
        else
        {
                ptr=start;
                while(ptr->link!=NULL && ptr->link->expo>ex)
                        ptr=ptr->link;
                tmp->link=ptr->link;
                ptr->link=tmp;
                if(ptr->link==NULL)  //item to be added in the end
                        tmp->link=NULL;
        }
        return start;
}
struct node *enter(struct node *start)
{
        int i,n,ex;
        float co;
        printf("How many terms you want to enter : ");
        scanf("%d",&n);
        printf("Enter each term with coeff and exp\n");
        for(i=1;i<=n;i++)
        {
                scanf("%f %d",&co,&ex);
                start=insert(start,co,ex);
        }
        return start;
}
struct node *poly_add(struct node *p1,struct node *p2)
{
        struct node *p3_start,*p3,*tmp;
        p3_start=NULL;
        if(p1==NULL && p2==NULL)
                return p3_start;

        while(p1!=NULL && p2!=NULL )
        {
                tmp= (struct node*)malloc(sizeof(struct node));
                if(p3_start==NULL)
                {
                        p3_start=tmp;
                        p3=p3_start;
```

```
		}
		else
		{
			p3->link=tmp;
			p3=p3->link;
		}
		if(p1->expo > p2->expo)
		{
			tmp->coef=p1->coef;
			tmp->expo=p1->expo;
			p1=p1->link;
		}
		else
			if(p2->expo > p1->expo)
			{
				tmp->coef=p2->coef;
				tmp->expo=p2->expo;
				p2=p2->link;
			}
			else
				if(p1->expo == p2->expo)
				{
					tmp->coef=p1->coef + p2->coef;
					tmp->expo=p1->expo;
					p1=p1->link;
					p2=p2->link;
				}
}
while(p1!=NULL)
{
		tmp= (struct node*)malloc(sizeof(struct node));
		tmp->coef=p1->coef;
		tmp->expo=p1->expo;
		if (p3_start==NULL) /*poly 2 is empty*/
		{
			p3_start=tmp;
			p3=p3_start;
		}
		else
		{
```

```
                p3->link=tmp;
                p3=p3->link;
            }
            p1=p1->link;
        }
        while(p2!=NULL)
        {
            tmp= (struct node*)malloc(sizeof(struct node));
            tmp->coef=p2->coef;
            tmp->expo=p2->expo;
            if (p3_start==NULL) /*poly 1 is empty*/
            {
                p3_start=tmp;
                p3=p3_start;
            }
            else
            {
                p3->link=tmp;
                p3=p3->link;
            }
            p2=p2->link;
}
        p3->link=NULL;
        return p3_start;
}
int main( )
{
        struct node *p1_start,*p2_start,*p3_start;
        p1_start=NULL;
        p2_start=NULL;
        p3_start=NULL;
        printf("Polynomial 1 :\n");
        p1_start=enter(p1_start);
        printf("Polynomial 2 :\n");
        p2_start=enter(p2_start);
        p3_start=poly_add(p1_start,p2_start);
        printf("Polynomial 1 is :  ");
        display(p1_start);
        printf("Polynomial 2 is :  ");
        display(p2_start);
```

```
        printf("Added polynomial is :  ");
        display(p3_start);
        return 0;
}
```

Input and Output Section:

```
Polynomial 1 :
How many terms you want to enter : 3
Enter each term with coeff and exp
1.2 5
2.5 4
4 3
Polynomial 2 :
How many terms you want to enter : 3
Enter each term with coeff and exp
2 6
1 2
1 0
Polynomial 1 is :  (1.2x^5) + (2.5x^4) + (4.0x^3)
Polynomial 2 is :  (2.0x^6) + (1.0x^2) + (1.0x^0)
Added polynomial is :  (2.0x^6) + (1.2x^5) + (2.5x^4) + (4.0x^3) + (1.0x^2)
+ (1.0x^0)
```

9. WAP to calculate factorial of a given no. (i) using recursion, (ii) using iteration.

Program:

```
#include <stdio.h>
#include<conio.h>
//Iterative function definition
long ifact(int n){
 long fact=1,i;
 for(i=1;i<=n;i++){
    fact=fact*i;
 }
 return fact;
}
//Recursion function definition
long rfact(int n){
   //base condition
```

```
    if(n<=1){
        return n;
    }
    // Recrsive Procedure
    else{
        return n*rfact(n-1);
    }
}
int main()
{
    int n;
    printf("Enter the number \n");
    scanf("%d",&n);
    //iterative function call
    printf("Iteration Method: %d Factorial is %ld",n,ifact(n));
    //recursion function call
    printf("\nRecursion Method: %d Factorial is %ld",n,rfact(n));
    getch();
    return 0;
}
```

Input and Output Section:

Enter the number
5
Iteration Method: 5 Factorial is 120
Recursion Method: 5 Factorial is 120
Enter the number
9
Iteration Method: 9 Factorial is 362880
Recursion Method: 9 Factorial is 362880

10. WAP to display Fibonacci series (i)using recursion, (ii) using iteration.

Program:

```
#include <stdio.h>
#include<conio.h>
//Iterative function definition
void ifib(int n){
   int a=0,b=1,c,i;
   if(n<1){
      printf("Fibonacci series : %d ",a);
   }
   else if(n==1){
      printf("Fibonacci series : %d %d ",a,b);
            }
      else{
         printf("Fibonacci series are: %d %d",a,b);
         for(i=2;i<=n;i++){
            c=a+b;
            printf(" %d",c);
            a=b;
            b=c;
         }
      }
   }
//Recursion function definition
int rfib(int n){
//base condition
   if(n==0){
      return 0;
   }
//second base condition
   else if(n==1){
      return 1;
   }
//Recursive Procedure
   else{
      return rfib(n-2)+rfib(n-1);
   }
}
int main()
```

```
{
   int n,result;
   printf("Enter the number \n");
   scanf("%d",&n);
   //iterative function call
   ifib(n);
   //recursion function call
   printf("\nFibonacci series are :");
   for(int i=0;i<=n;i++){
         printf(" %d",rfib(i));
        }
   getch();
   return 0;
}
```

Input and Output Section:

```
Enter the number
10
Fibonacci series are: 0 1 1 2 3 5 8 13 21 34 55
Fibonacci series are : 0 1 1 2 3 5 8 13 21 34 55

Enter the number
1
Fibonacci series : 0 1
Fibonacci series are : 0 1
```

11. WAP to calculate GCD of 2 number (i) with recursion (ii) without recursion.

Program:

```
#include <stdio.h>
#include<conio.h>
//Iterative function definition
int igcd(int n,int m){
   while(m!=n){
      if(m>n){
         m=m-n;
      }
      else{
         n=n-m;
      }
```

```
            }
return m;
    }
//Recursion function definition
int rgcd(int n,int m){
    //base condition
    if(n==m){
        return n;
    }
    // Recrsive Procedure
    if(m>n){
        return rgcd(m-n,n);
    }
        rgcd(m,n-m);
}
int main()
{
    int a,b;
    printf("Enter the two number \n");
    scanf("%d%d",&a,&b);
    //iterative function call
    printf("Iterative Method %d and %d GCD is %d ",a,b,igcd(a,b));
    //recursion function call
    printf("\nRecursion Method %d and %d GCD is %d",a,b,rgcd(a,b));
    getch();
    return 0;
}
```

Input and Output Section:

```
Enter the two number
13 117
Iterative Method 13 and 117 GCD is 13
Recursion Method 13 and 117 GCD is 13
Enter the two number
93 131
Iterative Method 93 and 131 GCD is 1
Recursion Method 93 and 131 GCD is 1
```

12. WAP to create a Binary Search Tree and include following operations in tree: (a) Insertion (Recursive and Iterative Implementation) (b) Deletion.

(a) Binary Search Tree Insertion Operation (Recursive and Iterative Implementation):

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct Node{
   struct Node *lchild;
   int data;
   struct Node *rchild;
}*root=NULL;
void Insert(int key)
{
   struct Node *t=root;
   struct Node *r,*p;
   if(root==NULL){
      p=(struct Node *)malloc(sizeof(struct Node));
      p->data=key;
      p->lchild=p->rchild=NULL;
      root=p;
      return ;
   }
   while(t!=NULL){
      r=t;
      if(key<t->data)
         t=t->lchild;
      else if(key>t->data)
         t=t->rchild;
      else
         return;
   }
      p=(struct Node *)malloc(sizeof(struct Node));
      p->data=key;
      p->lchild=p->rchild=NULL;
```

```
        if(key<r->data) r->lchild=p;
        else r->rchild=p;
}
    void Inorder(struct Node *p){
        if(p){
            Inorder(p->lchild);
            printf("%d ",p->data);
            Inorder(p->rchild);
        }
    }
struct Node * Search(int key){
    struct Node *t=root;
    while(t!=NULL){
        if(key==t->data)
            return t;
        else if(key<t->data)
            t=t->lchild;
        else
            t=t->rchild;
    }
    return NULL;
}

struct Node *RInsert(struct Node *p,int key){
    struct Node *t=NULL;
    if(p==NULL){
        t=(struct Node *)malloc(sizeof(struct Node));
        t->data=key;
        t->lchild=t->rchild=NULL;
        return t;
    }
        if(key<p->data)
            p->lchild=RInsert(p->lchild,key);
        else if(key>p->data)
            p->rchild=RInsert(p->rchild,key);

        return p;
}

int main(){
```

```
    struct Node *temp;
    printf("Iterative BST: \n");
    Insert(10);
    Insert(5);
    Insert(20);
    Insert(8);
    Insert(30);
    Inorder(root);
    printf("\n");

     temp=Search(30);
    if(temp!=NULL){
       printf("Element %d is Found \n",temp->data);
    }
    else
       printf("Element is not found \n");

    printf("Recursive BST: \n");
    root=RInsert(root,10);
    RInsert(root,5);
    RInsert(root,20);
    RInsert(root,8);
    RInsert(root,30);
    Inorder(root);
    printf("\n");
    temp=Search(20);
    if(temp!=NULL){
       printf("Element %d is Found \n",temp->data);
    }
    else
       printf("Element is not found \n");
    getch();
    return 0;
}
```

Input and Output Section:
Iterative BST:
5 8 10 20 30
Element 30 is Found
Recursive BST:
5 8 10 20 30
Element 20 is Found

(b) Binary Search Tree Deletion Operation (Recursive and Iterative Implementation):

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct Node{
   struct Node *lchild;
   int data;
   struct Node *rchild;
}*root=NULL;
void Insert(int key)
{
   struct Node *t=root;
   struct Node *r,*p;
   if(root==NULL){
      p=(struct Node *)malloc(sizeof(struct Node));
      p->data=key;
      p->lchild=p->rchild=NULL;
      root=p;
      return ;
   }
   while(t!=NULL){
      r=t;
      if(key<t->data)
         t=t->lchild;
      else if(key>t->data)
         t=t->rchild;
      else
         return;
```

```
    }
        p=(struct Node *)malloc(sizeof(struct Node));
        p->data=key;
        p->lchild=p->rchild=NULL;
        if(key<r->data) r->lchild=p;
        else r->rchild=p;
}
    void Inorder(struct Node *p){
        if(p){
            Inorder(p->lchild);
            printf("%d ",p->data);
            Inorder(p->rchild);
        }
    }
/*struct Node * Search(int key){
    struct Node *t=root;
    while(t!=NULL){
        if(key==t->data)
            return t;
        else if(key<t->data)
            t=t->lchild;
        else
            t=t->rchild;
    }
    return NULL;
}
*/
struct Node *RInsert(struct Node *p,int key){
    struct Node *t=NULL;
    if(p==NULL){
        t=(struct Node *)malloc(sizeof(struct Node));
        t->data=key;
        t->lchild=t->rchild=NULL;
        return t;
    }
        if(key<p->data)
            p->lchild=RInsert(p->lchild,key);
        else if(key>p->data)
            p->rchild=RInsert(p->rchild,key);
```

```
        return p;
}
int Height(struct Node *p){
   int x,y;
   if(p==NULL) return 0;
   x=Height(p->lchild);
   y=Height(p->rchild);
   return x>y?x+1:y+1;
}
struct Node *InSucc(struct Node *p){
   while(p && p->lchild!=NULL)
      p=p->lchild;
   return p;
}
struct Node *InPre(struct Node *p){
   while(p && p->rchild!=NULL)
      p=p->rchild;
   return p;
}
//BST Delete from a Element
struct Node *Delete(struct Node *p,int key){
   struct Node *q;
   if(p==NULL)
      return NULL;
   if(p->lchild==NULL && p->rchild==NULL){
      if(p==root)
         root=NULL;
      free(p);
         return NULL;
   }
   if(key<p->data)
      p->lchild=Delete(p->lchild,key);
   else if(key>p->data)
      p->rchild=Delete(p->rchild,key);
   else
      {
         if(Height(p->lchild)>Height(p->rchild)){
            q=InPre(p->lchild);
            p->data=q->data;
            p->lchild=Delete(p->lchild,q->data);
```

```
            }
            else{
                q=InSucc(p->rchild);
                p->data=q->data;
                p->rchild=Delete(p->rchild,q->data);
            }
        }
        return p;
}

int main(){

    printf("Iterative BST: \n");
    Insert(10);
    Insert(5);
    Insert(20);
    Insert(8);
    Insert(30);
    Inorder(root);
    printf("\nAfter Iterative BST Delete Element:  ");
    Delete(root,5);
    Inorder(root);
    printf("\n");


    printf("Recursive BST: \n");
    root=RInsert(root,10);
    RInsert(root,5);
    RInsert(root,20);
    RInsert(root,8);
    RInsert(root,30);
    Inorder(root);
    Delete(root,20);
    printf("\nAfter Recursive BST Delete Element:  ");
    Inorder(root);
    printf("\n");
    getch();
    return 0;
}
```

Input and Output Section:

Iterative BST:
5 8 10 20 30
After Iterative BST Delete Element: 8 10 20 30
Recursive BST:
5 8 10 20 30
After Recursive BST Delete Element: 5 8 10 30

13. WAP to create Binary Tree and display its pre-order, post-order and in-order traversals Recursively.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include "Queue.h"
#include<conio.h>
struct Node *root=NULL;
void Treecreate()
{
 struct Node *p,*t;
 int x;
 struct Queue q;
 create(&q,100);

 printf("Enter root value ");
 scanf("%d",&x);
 root=(struct Node *)malloc(sizeof(struct Node));
 root->data=x;
 root->lchild=root->rchild=NULL;
 enqueue(&q,root);

 while(!isEmpty(q))
 {
 p=dequeue(&q);
 printf("enter left child of %d ",p->data);
 scanf("%d",&x);
 if(x!=-1)
 {
 t=(struct Node *)malloc(sizeof(struct Node));
```

```
 t->data=x;
 t->lchild=t->rchild=NULL;
 p->lchild=t;
 enqueue(&q,t);
 }
 printf("enter right child of %d ",p->data);
 scanf("%d",&x);
 if(x!=-1)
 {
 t=(struct Node *)malloc(sizeof(struct
Node));
 t->data=x;
 t->lchild=t->rchild=NULL;
 p->rchild=t;
 enqueue(&q,t);
 }
 }
}
void Preorder(struct Node *p)
{
 if(p)
 {
 printf("%d ",p->data);
 Preorder(p->lchild);
 Preorder(p->rchild);
 }
}
void Inorder(struct Node *p)
{
 if(p)
 {
 Inorder(p->lchild);
 printf("%d ",p->data);
 Inorder(p->rchild);
 }
}
void Postorder(struct Node *p)
{
 if(p)
 {
```

```
 Postorder(p->lchild);
 Postorder(p->rchild);
 printf("%d ",p->data);
 }
}
int main()
{
 Treecreate();
 printf(" \nIn order ");
 Inorder(root);
 printf(" \nPre order ");
 Preorder(root);
 printf("\nPost Order ");
 Postorder(root);
 getch();
 return 0;
}
```

Queue Header File

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
 struct Node *lchild;
 int data;
 struct Node *rchild;
};
struct Queue
{
 int size;
 int front;
 int rear;
 struct Node **Q;
};
void create(struct Queue *q,int size)
{
 q->size=size;
 q->front=q->rear=0;
 q->Q=(struct Node **)malloc(q->size*sizeof(struct
Node *));
```

```
}
void enqueue(struct Queue *q,struct Node *x)
{
 if((q->rear+1)%q->size==q->front)
 printf("Queue is Full");
 else
 {
 q->rear=(q->rear+1)%q->size;
 q->Q[q->rear]=x;
 }
}
struct Node * dequeue(struct Queue *q)
{
 struct Node* x=NULL;

 if(q->front==q->rear)
 printf("Queue is Empty\n");
 else
 {
 q->front=(q->front+1)%q->size;
 x=q->Q[q->front];
 }
 return x;
}
int isEmpty(struct Queue q)
{
 return q.front==q.rear;
}
```

Input and Output Section:

```
Enter root value 10
enter left child of 10 20
enter right child of 10 30
enter left child of 20 40
enter right child of 20 50
enter left child of 30 60
enter right child of 30 70
enter left child of 40 80
enter right child of 40 90
```

```
enter left child of 50 -1
enter right child of 50 -1
enter left child of 60 -1
enter right child of 60 -1
enter left child of 70 -1
enter right child of 70 -1
enter left child of 80 -1
enter right child of 80 -1
enter left child of 90 -1
enter right child of 90 -1

In order 80 40 90 20 50 10 60 30 70
Pre order 10 20 40 80 90 50 30 60 70
Post Order 80 90 40 50 20 60 70 30 10
```

14. WAP to create Binary Tree and display its pre-order, post-order and in-order traversals Iteratively.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include "Stack.h"
#include "Queue.h"
#include<conio.h>
struct Node *root=NULL;
void Treecreate()
{
 struct Node *p,*t;
 int x;
 struct Queue q;
 create(&q,100);

 printf("Enter root value ");
 scanf("%d",&x);
 root=(struct Node *)malloc(sizeof(struct Node));
 root->data=x;
 root->lchild=root->rchild=NULL;
 enqueue(&q,root);

 while(!isEmpty(q))
```

```
 {
 p=dequeue(&q);
 printf("enter left child of %d ",p->data);
 scanf("%d",&x);
 if(x!=-1)
 {
 t=(struct Node *)malloc(sizeof(struct Node));
 t->data=x;
 t->lchild=t->rchild=NULL;
 p->lchild=t;
 enqueue(&q,t);
 }
 printf("enter right child of %d ",p->data);
 scanf("%d",&x);
 if(x!=-1)
 {
 t=(struct Node *)malloc(sizeof(struct
Node));
 t->data=x;
 t->lchild=t->rchild=NULL;
 p->rchild=t;
 enqueue(&q,t);
 }
 }
}
void IPreorder(struct Node *p){
        struct Stack stk;
        Stackcreate(&stk,100);
        while(p || !isEmpty(stk))
        {
                if(p){
                        printf("%d ",p->data);
                        push(&stk,p);
                        p=p->lchild;
                }
                else{
                        p=pop(&stk);
                        p=p->rchild;
                }
        }
```

```
}
void IInorder(struct Node *p){
        struct Stack stk;
        Stackcreate(&stk,100);
        while(p || !isEmpty(stk))
        {
                if(p){

                        push(&stk,p);
                        p=p->lchild;
                }
                else{
                        p=pop(&stk);
                        printf("%d ",p->data);
                        p=p->rchild;
                }
        }
}

int main()
{
 Treecreate();
 printf(" \nIn order ");
 IInorder(root);
 printf(" \nPre order ");
 IPreorder(root);

 getch();
 return 0;
}
```

Stack Header File

```
#include <stdio.h>
#include <stdlib.h>
//#include "Queue.h"
struct Stack
{
 int size;
 int top;
 struct Node **S;
```

```
};
void Stackcreate(struct Stack *st,int size)
{
 st->size=size;
 st->top=-1;
 st->S=(struct Node **)malloc(st->size*sizeof(struct Node *));
}

void push(struct Stack *st,struct Node *x)
{
 if(st->top==st->size-1)
 printf("Stack overflow\n");
 else
 {
 st->top++;
 st->S[st->top]=x;
 }

}
struct Node *pop(struct Stack *st)
{
 struct Node *x=NULL;
 if(st->top==-1)
 printf("Stack Underflow\n");
 else
 {
 x=st->S[st->top--];
 }
 return x;
}

int isEmpty(struct Stack st)
{
 if(st.top==-1)
          return 1;
 return 0;
}
int isFull(struct Stack st)
{
 return st.top==st.size-1;
```

```
}
```

Input and Output Section:

```
Enter root value 10
enter left child of 10 20
enter right child of 10 30
enter left child of 20 40
enter right child of 20 50
enter left child of 30 60
enter right child of 30 70
enter left child of 40 -1
enter right child of 40 -1
enter left child of 50 -1
enter right child of 50 -1
enter left child of 60 -1
enter right child of 60 -1
enter left child of 70 -1
enter right child of 70 -1

In order 40 20 50 10 60 30 70
Pre order 10 20 40 50 30 60 70
```

15. WAP to implement Diagonal Matrix using one-dimensional array.

Program:

```
#include<stdio.h>
#include<conio.h>
struct Matrix
{
 int A[10];
 int n;
};
void Set(struct Matrix *m,int i,int j,int x)
{
 if(i==j)
 m->A[i-1]=x;

}
int Get(struct Matrix m,int i,int j)
{
 if(i==j)
```

```
 return m.A[i-1];
 else
 return 0;
}
void Display(struct Matrix m)
{
 int i,j;
 for(i=0;i<m.n;i++)
 {
   for(j=0;j<m.n;j++)
   {
   if(i==j)
      printf("%d ",m.A[i]);
   else
      printf("0 ");
 }
   printf("\n");
 }
}
int main()
{
//clrscr();
 struct Matrix m;
 m.n=4;

 Set(&m,1,1,5);
 Set(&m,2,2,8);
 Set(&m,3,3,9);
 Set(&m,4,4,12);
// printf("%d \n",Get(m,3,3));
 Display(m);
 getch();
 return 0;
}
```

Input and Output section:

5 0 0 0
0 8 0 0
0 0 9 0
0 0 0 12

16. WAP to implement Lower Triangular Matrix using one-dimensional array.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct Matrix
{
 int *A;
 int n;
};
void Set(struct Matrix *m,int i,int j,int x)
{
 if(i>=j)
 m->A[m->n*(j-1)+(j-2)*(j-1)/2+i-j]=x;
}
int Get(struct Matrix m,int i,int j)
{
 if(i>=j)
 return m.A[m.n*(j-1)+(j-2)*(j-1)/2+i-j];
 else
 return 0;
}
void Display(struct Matrix m)
{
 int i,j;
 for(i=1;i<=m.n;i++)
 {
 for(j=1;j<=m.n;j++)
 {
 if(i>=j)
 printf("%d  ",m.A[m.n*(j-1)+(j-2)*(j-1)/2+i-j]);
```

```
 else
 printf("0   ");
 }
 printf("\n");
 }
}
int main()
{
 struct Matrix m;
 int i,j,x;
//clrscr();
 printf("Enter Dimension");
 scanf("%d",&m.n);
 m.A=(int *)malloc(m.n*(m.n+1)/2*sizeof(int));
 printf("enter all elements");
 for(i=1;i<=m.n;i++)
 {
 for(j=1;j<=m.n;j++)
 {
 scanf("%d",&x);
 Set(&m,i,j,x);
 }
 }
 printf("\n\n");
 Display(m);

getch();
 return 0;
}
```

Input and Output Section:

Enter Dimension
4
enter all elements
10 20 30 40
11 22 33 44
12 24 36 48
13 26 39 52

Lower Triangular Matrix are:

```
10   0    0   0
11  22    0   0
12  24  36   0
13  26  39  52
```

17. WAP to implement Upper Triangular Matrix using one-dimensional array.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct Matrix
{
 int *A;
 int n;
};
void Set(struct Matrix *m,int i,int j,int x)
{
 if(i<=j)
 m->A[m->n*(i-1)-(i-2)*(i-1)/2+j-i]=x;
}
int Get(struct Matrix m,int i,int j)
{
 if(i<=j)
 return m.A[m.n*(i-1)-(i-2)*(i-1)/2+j-i];
 else
 return 0;
}
void Display(struct Matrix m)
{
 int i,j;
 for(i=1;i<=m.n;i++)
 {
 for(j=1;j<=m.n;j++)
 {
 if(i<=j)
 printf("%d  ",m.A[m.n*(i-1)-(i-2)*(i-1)/2+(j-i)]);
 else
 printf("0   ");
```

```
 }
 printf("\n");
 }
}
int main()
{
 struct Matrix m;
 int i,j,x;
//clrscr();
 printf("Enter Dimension \n");
 scanf("%d",&m.n);
 m.A=(int *)malloc(m.n*(m.n+1)/2*sizeof(int));
 printf("enter all elements \n");
 for(i=1;i<=m.n;i++)
 {
 for(j=1;j<=m.n;j++)
 {
 scanf("%d",&x);
 Set(&m,i,j,x);
 }
 }
 printf("\n");
 printf("Upper Triangular Matrix are: \n");
 Display(m);

getch();
 return 0;
}
```

Input and Output Section:

```
Enter Dimension
4
enter all elements
10 20 30 40
11 12 13 14
21 22 23 24
31 32 33 34
```

Upper Triangular Matrix are:
10 20 30 40
0 12 13 14
0 0 23 24
0 0 0 34