1) Write 8051 ALP to move block with respect to internal RAM.

```
=>      ORG  0000H.
        MOV. R0, #40H ; load the 40H into R0.
        MOV  R1, #50H ; load the 50H value into R1.
        MOV  R2, #10H ; load the 10H value into R2.
BACK :  MOV. A, @R0   ; Move the data pointing R0 location to A.
        MOV @R1, A    ; Move the data from A to location of R1.
        INC R0        ; Increament the value of R0
        INC R1        ; Increament the value of R1
        DJNZ R2, BACK ; Decreament R2 and jump into BACK.
        END.
```

ALGORITHM:

Step1: Start from 0000H location
Step2: Move the value of 40H into a R0.
Step3: Move the value of 50H into a R1.
Step4: Move the value of 10H into R2.
Step5: When loop is activated, move the data pointing to R0 location into A.
Step6: Move the data from A into location of R1.
Step7: Increament R0 and R1.
Step8: Decrement R2 and jump Back to loop. if R2 is not Equal to zero.
Step9: END.

Result:

Before Execution:

Addressing Mode.		R0	R1	R2	A	
Value.		00X00H	00X00H	00X00H	00X00H.	
Input's:	D:00X40H	10	20	30	40	50
	D:00X50H	00	00	00	00.	00

After execution:

Addressing Mode		R_0	R_1	R_2	A	
Value		00X50H	00X60H	00X00H	00X50H	
Output:	D: 00X40H	00	00	00	00	00
	D1 00X50H	10	20	30	40	50

2) Write 8051 ALP to move block with respect to external RAM

⇒

```
ORG 0000H
MOV DPTR, #2000H ; load the 2000H value in DPTR.
MOV R0, #30H. ; load the 30H value in R0.
MOV A, @R0 ; move the data from R0 location to A.
MOV R2, #09H ; move the value of 09H into R2.
Loop : MOVX @DPTR, A ; move the data from A to location of DPTR
INC R0 ; increament R0.
MOV A, @R0 ; move the data from R0 location to A.
INC DPTR ; increament DPTR.
DJNZ R2, LOOP ; decrament R2 and jump back to loop.
END.
```

ALGORITHM:

Step 1: Start from 0000H location.
Step 2: Move the 2000H value into DPTR.
Step 3: Move the value of 30H into R_0
Step 4: Move the data of 09H into R_2.
Step 5: Move the data from location of R_0 to A.
Step 6: When loop is activated, move the data pointing to A
Step 7: Increament R_0.
Step 8: Move the data from location of R_0 to A.
Step 9: Increament DPTR.
Step 10: Decrement R_2 and jump back to loop
Step 11: Stop.

Result:

Before Execution:

Input:	D:00x30H	10	20	30	40	50	60	70	80	90	99
	X: 00x2000H.	00	00	00	00	00	00	00	00	00	00

After Execution:

output:	D:00x30H.	00	00	00	00	00	00	00	00	00	00
	X:00x2000H	10	20	30	40	50	60	70	80	90	99

3) Block Exchange with respect to internal RAM. For this. Write 8051 ALP.

⇒
```
        ORG   0000H.
        MOV   R0, #60H ; load the value of 60H into R0.
        MOV   R1, #70H ; load the value of 70H into R1.
        MOV   R2, #10H ; load the value of 10H into R2.
LOOP:   MOV   A, @R0   ; Move the data pointing from R0 to A
        XCH   A, @R1   ; Exchange the data between A and location of R
        MOV   @R0, A   ; move the data of A into location of R0.
        INC   R0       ; Increament R0.
        INC   R1       ; Increament R1.
        DJNZ  R2, LOOP ; decreament R2 and jump back to the LO
        END            ; END
```

ALGORITHM:

Step1: Start from 0000H location

Step2: load the value of 60H into R0.

Step3: move the value of 70H into R1.

Step4: move the value of 10H into R2.

Step5: When the loop gets activated, move the data pointing to R0 into A.

Step6: Exchange data between A and R1 location.

Step7: Again move the data from A to R0 location

Step8: Increament R0 and R1

Step 9: decreament R_2 and jump to the loop if R_2 is not equal to zero.

Step 10: END.

Results:

Before Execution:

Input: D:00X60H. 10 20 30 40 50 60 70 80 90 99

D:00X70H 11 21 31 41 51 61 71 81 91 98

After Execution:

Output: D:00X60H 11 21 31 41 51 61 71 81 91 98

D:00X70H 10 20 30 40 50 60 70 80 90 99

4) Write 8051 ALP to exchange a block of data with respect to external RAM

⟹

```
ORG 0000H
MOV DPTR, #2000H ; load the value of 2000H into DPTR.
MOV R0, #30H     ; load the 30H data into R0.
MOV R2, #10H.    ; Move the data of 10H into R2.
Loop: MOVX A, @DPTR.  ; Move the data from DPTR location to A.
XCH A, @R0.      ; Exchange data between A and location R0.
MOVX @DPTR, A    ; move the data from A to DPTR location
INC DPTR         ; Increament DPTR.
INC R0           ; Increament R0.
DJNZ R2, LOOP.   ; decreament R2 and jump back to loop
END.
```

ALGORITHM:

Step 1: Start from 0000H location.

Step 2: move the data of 2000H into DPTR.

Step 3: move the data of 30H into R0.

Step 4: move the data of 10H into R2.

Step 5: When loop is activated, move the data from location of DPTR to Accumulator(A).
Step 6: Exchange the data between A and location of R_0.
Step 7: Again move the data from A to location of DPTR
Step 8: Increament DPTR and R_0 value.
Step 9: decrement R_2 and jump back to loop if R_2 is not Equal to zero.
Step 10: END.

Results:

Before Execution:

Inputs: D: 00x30H 09 08 07 06 05 04 03 11 12 13
X: 00x2000H 21 22 23 24 25 26 27 28 29 30

After Execution:

Output: D: 00x30H 21 22 23 24 25 26 27 28 29 30
X: 00x2000H 09 08 07 06 05 04 03 01 12 13

5) Write 8051 ALP to add N 16,8 bit numbers.

⇒
```
ORG 000H.
MOV R0, #20H. ; load the value of 20H into R0.
MOV R2, #04H. ; load the value of 04H into R2.
LOOP: MOV A, @R0. ; move the data from location of R0 to A.
INC R0. ; Increament R0.
ADDC A, @R0. ; Add the data's from A and location of R0.
DJNZ R2, LOOP. ; decreament R2 and jump back to loop
END.
```

ALGORITHM

Step 1: Start from 0000H location.
Step 2: move the data from of 20H into R_0 to indicate location.
Step 3: move the value of 04H into R_2 to indicate count.

Step 3: When loop is activated, move the data from location of R0 to A.
Step 4: Increament R0 value.
Step 5: Then add data from A and location of R0.
Step 6: decreament R2 and jump back to loop if R2 is not equal to zero.
Step 7: END.

Result:
Before Execution:
Input: D:00X20H. 21 22 64 88

After Execution:
Output: Adressing mode R0 R2 A
Value. 00X24H. 00X00H. 00X69H

6) Write 8051 ALP to Substract N, 8, 16 and 32 bit numbers.
⇒ ORG 000H
MOV R0, #20H. ; load the value of 20H into R0.
MOV R2, #04H. ; load the value of 04H into R2
MOV A, @R0. ; move the data from location of R0 to A.
LOOP: MOV R3, A ; move the data from A to R3.
INC R0. ; Increament R0.
MOV A, @R0. ; move the data from location of R0 to A.
SUBB A, R3. ; Substract the values from A and R3.
DJNZ R2, LOOP. ; decreament R2 and jump back to loop.
END.

ALGORITHM:
Step 1: Start from 0000H location
step 2: move the value of 20 H into R0.
Step 3: move the value of 04 H into R2 to indicate count's.

Step 4: move the data from location of R_0 to A in loop.
Step 5: move the data from A to Register R_3.
Step 6: Increament R_0 Value.
Step 7: Again move the data from location of R_0 to A.
Step 8: Substraction of values takes between A and R_3 value
Step 9: Then decreament R_2 and jump back to loop if R_2 is not Equal to zero.
Step 10: END.

Results:

Before Execution:

Input: D:00X20H 98 12 09 08 10.

After Execution:

Output: Addressing mode R_0 R_2 R_3 A
Value 00X25H. 00X00H. 00X10H. 00X4BH

7) Write 8051 ALP to multiplication of N 8bit numbers.

⇒
```
        ORG 0000H
        MOV R0,#20H ; load the value of 20H into R0.
        MOV R2,#04H ; load the value of 04H into R2
        MOV A,@R0   ; move the data from location of R0 to A.
LOOP:   INC R0      ; Increament R0.
        MOV B,@R0   ; Again move the data from location of R0 to
        MUL AB      ; multiply AB.
        DJNZ R2,LOOP ; decreament R2 and jump back to loop.
        END
```

ALGORITHM:

Step 1: Start from 0000H location
Step 2: move the value of 20H into R_0 to indicate location.
Step 3: move the value of 04H into R_2 to indicate count.

Step 4: move the data from location of R_0 to A
Step 5: When loop is activated, increament R_0.
Step 6: Again Move the data from location of R_0 to B.
Step 7: multiply A and B.
Step 8: Dcreament R_2 and jump back to the loop if R_0 is not Equal to zero.

Results:

Before Execution:

Input: D:00X20H. 10 11 20 21 09

After Execution:

Output:

Addressing Mode	R_0	R_2	B.	A
Value.	00X24H.	00X00H.	00X09H	00X65823H

8. Write 8051 ALP to divide 8 bit Number.

```
=>      ORG 0000H.
        MOV DPTR,#1000H. ; load the value of 1000H into DPTR.
        MOVX A,@DPTR.    ; move the data from location of DPTR to A
        MOV B,#10H.      ; load the value of 10H into B.
        DIV AB.          ; divide A by B.
        END.
```

ALGORITHM.

Step 1: Start.
Step 2: Move the value of 1000H into DPTR. To indicate location.
Step 3: Move the data from DPTR location to A.
Step 4: Move the value of 10H into B.
Step 5: Devide A by B. and Store the result in A.
Step 6: END.

Page No :
Date :

Result:

Before Execution:

Input: Addressing Mode A B R0 X':00x2000H

Value 00x00H. 00x00H. 00x00H. 20x20H.

After Execution:

Output: Addressing Mode A B R0 X':00 x1000H.

Value 00x02H. 00x00H. 00x00H. 00x00H.

9. Write 8051 ALP to Find Average of N numbers.

⇒ ORG 000H.

MOV R0, #20H ; Move the value of 20H into R0.

MOV R2 #05H. ; load the value of 05H into R2.

loop: ADDC A, @R0. ; Add values from A and location of R0.

INC R0. ; Increament R0.

DJNZ R2, loop; Decreament R2 and jump back to loop.

MOV B, #05H. ; load the value of 05H into B.

DIV AB. ; Devide A by B.

END.

ALGORITHM

Step 1: Start.

Step 2: Move the value of 20H into R0 to indecate location.

Step 3: Move the value of 05H into R2 to indecate count of R2.

Step 4: When loop is activated, Add the data from location of R0 to A.

Step 5: Increment R0 value.

Step 6: Decrement R2 and jump back to loop if R2 is not equal to zero.

Step 7: load the value of 05H into B. which indicate total numbers.

Step 8: Devide A by B.

Step 9: END.

Result:

Before Execution:

Input : D:00x20H. 10 20 30 40 50.

After Execution:

Output: Addressing Mode R0 A B

Value. 00x24H. 00x30H. 00x00H.

10. Write 8051 ALP to Find factorial of a Number.

⟹ ORG 0000H.

```
    MOV R0,#05H.   ; load the value of 05H into R0
    MOV A,R0.      ; Move the data from R0 to A.
LOOP: DEC R0.      ; decrement R0.
    MOV B,R0.      ; move the data from R0 to B.
    MUL AB.        ; multiply A and B.
    CJNE R0,#01H,LOOP.; Compare R0 with 01H value and jump
    END.                  to LOOP.
```

ALGORITHM:

Step 1: Start from 0000H location.

Step 2: Move the value of 05H into R0.

Step3: Move the data from R0 to A.

Step4: When LOOP is activated, decreament R0

Step5: Again move the data from decremented R0 to B.

Step6: Multiply A and B.

Step7: Compare the value of R0 and 01H. if they are not equal jump back to loop.

Step8: END.

Result:

Before Execution:

Addressing Mode R0 A B

Value 00x00H 00x00H 00x00H.

After Execution:

Addressing Mode	R_0	A	B
Value.	00X05H	00X120H.	00X01H.

11. Write 8051 ALP to Positive or Negative number from given array of N no's.

⇒

```
ORG 0000H.
    MOV R0, #20H.   ; load the 20H into R0.
    MOV R1, #05H.   ; load the 05H into R1.
LOOP: MOV A,@R0.    ; Move the data from location of R0 to A.
    INC R0.         ; Increment R0.
    RLC A           ; Rotate A left with carry.
    JC NEG.         ; Jump to NEG if carry bit is set.
    INC R4.         ; Otherwise increment R4.
    DJNZ R1, LOOP.  ; decrement R1 jump back to loop.
    SJMP DOWN.      ; Short jump to down.
NEG : INC R3.       ; In NEG loop increment R3.
    DJNZ R1, LOOP.  ; decrement R1 jump back to loop.
DOWN : NOP          ; No operation
    END.
```

ALGORITHM:

Step 1: Start.

Step 2: Move the data from 20H into R0 to indicate location.

Step 3: Move the data of 05H into R1 to indicate N numbers.

Step 4: When loop is activated, move the data from location of R0 to A.

Step 5: INCREMENT R0 Value.

Step 6: Rotate A to the left with Carry.

Step 7: Jump if Carry bit set, to NEG loop. Otherwise increment R4 for Positive Count.

Step 8: IN NEG loop increment R3 for Negative count.

Step 9: decrement R_1 and jump back to loop if R_1 is not Equal to zero.

Step 10: If the loop is completed at positive count. then Short jump to DOWN.

Step 11 : END.

Results:

Before Execution:

Input: D:00X20H : 10 -11 -21 14 05

After Execution:

Output:

Addressing Mode	R_0	R_1	A	R_3	R_4
Value.	00X24H.	00X00H.	00X10H.	00X02H	00X03H.

12. Write 8051 ALP to find Even or odd number from given array of N numbers.

⇒

```
        ORG  0000H.
        MOV R0,#20H.  ; load the value of 20H into R0
        MOV R1,#05H.  ; load the value of 05H into R1
LOOP:   MOV A,@R0     ; move the data from location of R0 to A.
        INC R0        ; Increment R0.
        RRC A         ; Rotate A to the Right with Carry.
        JC ODD.       ; If Carry bit is set then jump to ODD
        INC R4.       ; Otherwise increment R4. for even count.
        DJNZ R1, LOOP; decrement R1 and jump back to LOOP
        SJMP DOWN     ; Short jump to DOWN.
ODD:    INC R3        ; IN ODD loop. increment R3 for odd count
        DJNZ R1, LOOP ; decrement R1 and jump back to LOOP.
DOWN:   NOP           ; No operation.
        END
```

Page No.
Date

Result:

Before Execution:

Input: D: 00x20H. 11 12 13 14 16.

After Execution:

output: Addressing Mode R0 R1 A R3 R4

Value. 00x20H 00x00H. 00x16H 00x02H 00x03H.

13. Write 8051 ALP to find 0's and 1's count in a given number

=> ORG 0000H.

```
        MOV R0,#20H.  ; Load the value of 20H into R0
        MOV R2,#08H.  ; Load the value of 08H into R2
LOOP:   MOV A,@R0     ; When loop is activated, move data from R0 to A
LOOP:   RRC A         ; Rotate A to Right with Carry.
        JC ONE        ; If Carry bit is set jump to ONE loop.
        INC R4.       ; Otherwise increment R4. for zero count.
        DJNZ R2,LOOP. ; decrement R2 and jump back to loop.
ONE:    INC R3        ; In ONE loop increment R3 for ONE count
        DJNZ R2,LOOP. ; decrement R2 and jump back to Loop.
        END.
```

ALGORITHM

Step1: Start.

Step2: Load the 20H into R0. for location indication

Step3: Load the 08H into R2 for count.

Step4: When loop is activated, move data from location of R0 to A.

Step5: In loop. Rotate A to Right with Carry.

Step6: If Carry bit is set jump to ONE loop, thin increment R3 for one's count. Otherwise increment R4 for zero's count.

Step7: decrement R2 and jump back to loop if R2 is not equal. zero.

Step8: END.

Result:

Before Execution:

Input: 0:00x20H. 08

After Execution:

Addressing Mode	R_0	R_2	R_3	R_4
Value	00x20H	00x00H	00x01H	00x07H.

Page No.:
Date :

Name - Sukanya Nadagadalli
DIV - B
ROLL NO - 123.

1) Write a 8051 C- code to send data 48H to P0, P1, P2, P3 continously.

→ Algorithm:

① send 48H (or) initialise 48H data to P0, P1, P2, P3.
② To work it continously, put it into a while infinite loop.
③ Go to ports, one we will see that 48H is stored continously.

Program:

```
#include "reg51.h"
void main()
{
    while(1)                    TO store value continously.
    {   P0= 0x48H;   store 48H in Port 0
        P0 = 0x48H;  store 48H in Port 1
        P2 = 0x48H;  store 48H in Port 2
        P3 = 0x48H;  store 48H in Port 3.
    }
}
```

Result:

P0 : 48H
P1 : 48H
P2 : 48H
P3 : 48H.

a) Write a 8051 c-program to send data 00H and FFH alternatively to P0, P1, P2, P3 continously.

→ Algorithm:-

1. Initialize all ports P0, P1, P2, P3 to zero.
2. From zero to FF, get a for loop and store P0, P1, P2, P3 to FF.
3. The FOR loop will work as a delay element ov

Program:-

```
#include<reg51.hy
void main()
{
   while(1)                    // Infinite loop.
   {
     unsigned char i;
     P0 = 0;     store 0 in P0
     P1 = 0;     store 0 in P1
     P2 = 0;     store 0 in P2
     P3 = 0;     store 0 in P3
     for(i=0; i<=255; i++); Provides delay.
     P0 = 0xff;  store FF in P0
     P1 = 0xff;  store FF in P1
     P2 = 0xff;  store FF in P2
     P3 = 0xff;  store FF in P3
   }
}
```

3) Write a 8051 c- code to send single bit '0' and '1' to P0.0 continowly.

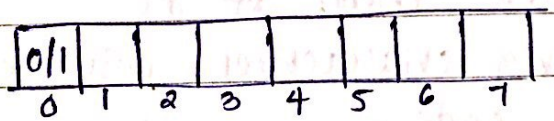
→ Algorithm -

1. Initidlize the P0.0 as a sbit of x.
2. In main, put a infinite while loop.
3. Intialize x to 0 and x to 1
4. End.

Program :-

```
#include <reg51.hy
sbit X = P0^0;
void main()
{
    while(1)
    {
      x=0;
      x=1;
    }
}
```

Result :-



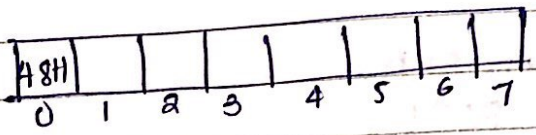
4) Write a c-program to send data 48H seriallY to P0.0 continoubly.

→ Algorithm -

1. Start
2. Initialisee P0.0 sbit of x.
3. Put, a infinite nonile loop.
4. Store data 48H in P0

Program –

```
#include <reg51.h>
sbit x = P0^0;  Initialize P0.0 to x.
void main()
{
    while(1)
    {  x=48H;   store 48H data at location of x.
    }
}
```

Result :- 

5) Write 8051 c-program to convert 0, 1, 2, 3, 4, 5, A, B, C, D characters (key) equivalent ASCII code.

→ Algorithm :-

1. Start
2. Store that given string
3. Take a for loop with the given size of string, and store them at P0.
4. The string character will be internally stored in ASCII code.
5. End.

Program :-

```
#include <reg51.h>
void main()
{   unsigned char num[] = "012345ABCD";  store the
    unsigned char Z;
```

for (z=0 ; z<=10 ; z++).

PO = num[z] ; store string value at PO.

}

Result:- PO

30	31	32	33	34	35	41	42
0	1	2	3	4	5	6	7

.

6) Write a 8051 C-program to toggle P1 with 1msec time delay.

→ Algorithm -

1. Start
2. Store 55 H in Port 3
3. Provide a delay, by setting timer High and Time Low to 00, Timer Run to 1. and if Timer flay is 1 reset it to 0. And also Timer Run to 0, after the execution time.
4. Store P3 as AA H.

Program:-

```
#include <reg51.hy
void main()
{
   unsigned char i;
   P3 = 0x55 ; store 55H in Port 3.
   delay(); Function call.
   P3 = 0xAA ; store AAH in Port 3.
}
void delay()
{   unsigned char i;
```

```
TMOD = 0x00;
for (i=0; i<14; i++)
{
    TMOD = 0x00; store 00 to Timer Mode.
    TH0 = 0x00; store timer high as 0.
    TL0 = 0x00; store timer low as 0
    TR0 = 1; Timer Run as 1.
    while(TF0==0); provide delay till timer flag is 0.
    TF0=0; Timer flag as 0.
    TR0 = 0; store timer Run as 0.
}
}
```

Result :- P3 :

55	55	55	55	55	55	55	55
0	1	2	3	4	5	6	7

P3 :

AA	AA	AA	AA	AA	AA	AA	AA
0	1	2	3	4	5	6	7.

7) Write 8051 c- program to generate square wave if T_{ON} = 2s and T_{OFF} = 1s.

→ Algorithm :-

1. Start
2. Initialise P0 as FF i.e High.
3. Provide a delay of 2s., with for loop till 2sec
4. store 00 in P0 i.e low.
5. Provide a delay of 1sec, with for loop till 14 sec.
6. END.

Program :-

```
#include <reg51.hy.
void delay2();   Function declaration
void delay1();   Function declaration.
void main()
{
   unsigned char i;
   while(1)          Infinite loop.
   {
      P0 = 0xFF;    store FF value in P0
      delay2();     Function call
      P0 = 0x00;    store 00 value in P0
      delay1();     function call.
   }
}
void delay2()
{
   unsigned char i;
   for(i=0; i<=28; i++)
   {
      TMOD = 0x01;  store 01 in TMOD
      TH0 = 0x00;   store 00 in TH0
      TL0 = 0x00;   store 00 in TL0
      TR0 = 1;      store 1 in TR0
      while(TF0==0); delay.
      TF0 = 0;      store 0 in TF0
      TR0 = 0;      store 0 in TR0.
   }
}
```

```
void delay1 ()
{
    unsigned char i;
    for (i=0 ; i<= 14 ;i++)
    {
        TMOD = 0X01 ; store 01 in TMOD
        TH0 = 0X00 ; store 00 in TH0
        TL0 = 0X00 ; store 00 in TL0
        TR0 = 1 ;    store 1 in TR0
        while (TF==0) ; provides delay.
        TF=0 ;  store 0 in TF
        TR0=0 ; store 0 in TR0.
    }
}.
```

Result :-

P0

FF	FF	FF	FF	FF	FF	FF	FF
0	1	2	3	4	5	6	7

P0

00	00	00	00	00	00	00	00
0	1	2	3	4	5	6	7

.

8) Write a 8051 - cprogram to generate Sawtooth wave

→ Algorithm -

1. Start
2. Take a loop from 0 to 255, which increases linearly with a delay.
3. Take another for loop from 255 to 0, which decreases linearly with a delay.
4. End.

Program -

```
#include <reg51.hy
void delay();        Function declaration.
void main()
{
   unsigned char i;
   while(1)                    Infinite loop.
   { for (i=0; i<=255; i++)
      {
         P1 = i;   store value of i in P1.
         delay();  function call.
      }
      for (i=255; i>=0; i--)
      {
         P1 = i;   store the value of i in P1
         delay();  function call.
      }
   }
}
Void delay()
{
   unsigned char i;
   for (i=0; i<=14; i++)
   {
      TMOD = 0x01;  store 01 in TMOD
      TH0 = 0x00;   store 00 in TH0
      TL0 = 0x00;   store 00 in TL0
      TR0 = 1;      store 1 in TR0
      while (TF0==0); time delay
      TF0 = 0;      store 0 in TF0
      TR0 = 0;      store 0 in TR0.
   }
}
```

9) Write a 8051 c-program to generate sine wave.

→ Algorithm -

1. Start
2. Take an array of different values of sine values.
3. In an infinite loop, store that array values in Port 1, till it reaches array size
4. End.

Program:-

```
#include <reg51.h>
void main()
{ unsigned int sinedata[] = {127, 148, 168, 187, 205
        220, 233, 243, 250, 253, 253, 250, 243,
        220, 205, 187, 168, 148, 127, 106, 86, 66,
        49, 34, 21, 11, 4, 0, 4, 11, 21, 34, 49, 66,
        86, 106, 127}; initialise data
  unsigned int i;
  while(1)
  {
     for (i=0; i<=38; i++)
          P1 = sinedata[i]; store data array in P0
  }
}
```

10) Write a c-program 8051 to blink led 16 & led

→

```
#include "at89c51xda.h"
void delay();
sbit led1 = P0^0; Initialise led1 at P0.0
sbit led2 = P0^3; Initialise led2 at P0.3.
```

Page No.
Date

```
void main()
{
   while(1)
   {
      led1=0;   led1 is stored as 0
      led2=0;   store 0 in led2
      delay();  function call
      led1=1;   store 1 in led1
      led2=1;   store 1 in led2.
   }
}
void delay()
{
   unsigned char i;
   for(i=0; i<14; i++)  To run loop.
   {
      TMOD=0x00;  store 00 in TMOD
      TH0=0x00;   store 00 in TH0
      TL0=0x00;   store 00 in TL0
      TR0=1;      store 1 in TR0
      while(TF0==0); delay.
      TF0=0;      store 0 in TF0
      TR0=0;      store 0 in TR0.
   }
}
```

Algorithm -

1. Start
2. Initialise led1 & led2.
3. To ON led, store 00 in it.
4. Provide Delay.
5. To OFF led, store 1 in it.

11) Write 8051 c-program to blink LED using sw

→ Algorithm -

1. Start
2. Initialise Led1 and switch as SW.
3. If SW. is equal to 0, then Led17 will glow
4. When, SW is equal to 1, then Led17 will be off
5. Stop.

Program -

```
#include "at89c51xd2.h"
sbit led17 = P0^0;   led17 is stored at P0.
sbit sw1 = P1^0;     SW is stored at P1.
void main()
{
    if (SW==0)        If switch is ON.
        led17 = 0;    store 0 in P0
    else
        led17 = 1;    store 1 in Led17.
}
```

12) Write a 8051 c-program to display 7-segment display

→ Algorithm -

1. Start.
2. Store the segment Hexadecimal value in respective Port.
3. Store 00 in P0

(4) store 3F in P2
(5) store 06 in P2
(6) store 5B in P2
(7) store 4F in P2.
(8) END.

Program -

```
#include "at89c51xd2.h"
sbit P0^0 = 0x00; store 00 in P0
void main()
{
   while(1)
   {  P2 = 0x3F; store 3F value in P2.
      delay();
      P2 = 0x06; store 06 value in P2
      delay();
      P2 = 0x5B; store 5B value in P2
      delay();
      P2 = 0x4F; store 4F value in P2.
      delay();
   }
}
void delay()
{  unsigned int i;
   for(i=0; i<=45000; i++); delay provided.
}
```

3) Write 8051 program to interface steeper motor, to rotate one complete cycle clockwise.

Date :

→ Algorithm -

1. Start
2. store 01 in P0 , provide delay
3. store 02 in P0 , provide delay.
4. store 04 in P0 , provide delay.
5. store 08 in P0 , provide delay.
6. stop.

Program -

```
#include "at89c51xda.h"          void delay();
void main()
{
    unsigned int i;
    while(1)
    {   for(i=0; i<=20; i++)
        {
            P0 = 0x00;   store 00 in P0
            delay();
            P0 = 0x 02;  store 02 in P0
            delay();
            P0 = 0x 04;  store 04 in P0
            delay();
            P0 = 0x 08;  store 08 in P0.
            delay();
        }
    }
}
void delay()
{
    unsigned int i;
    for(i=0; i<=15000; i++); delay is Provid
}
```

14) Write a 8051 C- program to interface 7-segment display, and design up-counter to display from 0 to 9.

→ Algorithm -

① start
② store 00 in P0, store 7D in P2,
③ store 3F in P2, store 07 in P2,
store 06 in P2, store 7F in P2,
store 4F in P2, store 67 in P2.
store 66 in P2,
store 6D in P2,
④ Provide a Delay of 1sec after each elemene store in P2.
⑤ stop.

Program -

```
#include "at89c51xda.h"
void delay()
{ unsigned int i;
  for( i=0 ; i<= 45000 ; i++) ; Provides delay.
}
void main()
{  P0= 0x00 ;
   while(1)
   {  P2 = 0x3F ; store 3F in P2
      delay() ;
      P2 = 0x06 ; store 06 in P2
      delay() ;
      P2 = 0x4F ; store AF in P2.
      delay()
```

```
P2 = 0X66H ;   Store 66H in P2
delay();
P2 = 0X6DH ;   store 6DH in P2
delay();
P2 = 0X7DH ;   store 7D in P2
delay();
P2 = 0X07H ;   Store 07 in P2
delay();
P2 = 0X67H ;   store 67 in P2.
delay();
} P
}
```

15) write a 8051 c-program to interface stepper motor
When you press switch-1 ⇒ stepper motor rotates clockwise.
When you press switch-2 ⇒ stepper motor rotates Anticlockwise.

→ Algorithm -

1. start
2. Store SW1 in P0.0 and store SW2 in P0.1.
3. If SW1 = 0, then store 01 in P1
   02 in P1
   04 in P1
   08 in P1.
4. If SW1 = 1, then store 08 in P1
   04 in P1
   02 in P1
   01 in P1
5. END.

Page No.:
Date:

Program :-

```
#include "at89c51xda.h"
sbit SW1 = P0^0;
sbit SW2 = P0^1;
void delay();
void main()
{
  unsigned int i;
  while(1)
  {
    if (SW1==0)   compare SW1 if 0
    {
      for(i=0; i<=20; i++)
      {
        P1 = 0x01;  store 01 in P1
        delay();
        P1 = 0x02;  store 02 in P1
        delay();
        P1 = 0x04;  store 04 in P1
        delay();
        P1 = 0x08;  store 08 in P1.
        delay();
      }
    }
    else if (SW1==1)   compare SW1 if 0
    {
      for(i=0; i<=20; i++)
      {
        P1 = 0x08;  store 08 in P1
        delay();
        P1 = 0x04;  store 04 in P1
        delay()
```

```
P1 = 0x02 ; store 02 in P1
delay() ;
P1 = 0x01 ; store 01 in P1.
delay() ;
}
}
}
}
void delay()
{
unsigned int i ;
for ( i = 0 ; i <= 15000 ; i++) ; delay to Provide.
}.
```

16) Write a 8051 C- program to interface LCD, to display RVBCET on the screen.

→ Algorithm -