8LEDs

- --> led is off
- * --> led is on

1. Blink the one led continuously others should be off.
2. BLink the one led five times continuously and then stop.
3. Create your own up counter using four leds.
4. Create your own down counter using four leds.
5. Create your own up counter using eight leds.
6. Create your own down counter using eight leds.
7. Blink led in following pattern.
    a. - - - -
    b. * - - -
    c. - * - -
    d. - - * -
    e. - - - *
8. Blink led in following pattern.
    a. - - - -
    b. - - - *
    c. - - * -
    d. - * - -
    e. * - - -
9. Blink led in following pattern.
    a. - - - -
    b. * * * *
10. Blink led in following pattern.
    a. * - * -
    b. - * - *
11. Blink led in following pattern.
    a. - - - -
    b. * - - -

c. - * - -
d. - - * -
e. - - - *
f. - - - -
g. - - - *
h. - - * -
i. - * - -
j. * - - -

Switches

1. One switch and one led.
2. Two switches and two leds.
3. One switch and one led(toggling).
4. One switch and four leds led(up counter).
5. One switch and four leds led(down counter).
6. Two switches and four leds(one switch for up counter and one for down counter).

Seven segment

1. Create an up counter (0-9) using a seven segment display.
2. Create a down counter(9-0) using a seven segment display.
3. Up counter using switch(One switch).
4. Down counter using switch(One switch).
5. Toggling up and down counter (One switch);

6. Toggling up and down counter(Two switches).
7. Display any two digits number using single port.
8. Two digit up counter.
9. Two digit down counter.
10. Two digit up counter using switch.
11. Two digit down counter using switch.
12. Two digit Up and down counter using two switches.
13. Three digit display using single port.
14. Three digit up counter.
15. Three digit down counter.
16. Three digit up counter using switch.
17. Three digit down counter using switch.
18. Three digit up and down counter using two switches.
19. Four digit display using single port.
20. Four digit up counter.
21. Four digit down counter.
22. Four digit up counter using switch.
23. Four digit down counter using switch.
24. Four digit up and down counter using two switches.

Liquid crystal Display(LCD)

1. Display any ascii character in the Lcd screen.
2. Display ‘a’ to ‘z’ in a continuous manner.
3. Display ‘z’ to ‘a’ in a continuous manner.
4. Display ‘ 0’ to ‘9 ’ in a continuous manner.
5. Display ‘9’ to ‘0’ in a continuous manner.
6. Display a string in the screen.
7. Display a moving string left to right.
8. Display a moving string right to left.
9. Display 0 to 99 up counter in the Lcd screen.

10. Up counter in lcd using switch.
11. Swastik symbol.
12. Standing person symbol.
13. Lifting hands and standing symbol.
14. Heart symbol.
15. Arrow mark.
16. Battery with one slice.
17. Battery with two slice.
18. Battery with three slice.
19. Network tower symbol.
20. Heart symbols using four pixels.
21. Beating heart.
22. Person standing and lifting his hand.
23. Winking smiley.
24. Arrow symbol moving from left to right.
25. Arrow symbol moving from right to left.
26. Charging battery symbol.
27. Charging battery symbol with percentage.

Keypad

1. Create a numerical keypad and display it in seven segment.
2. Create a numerical keypad and display it in the lcd.
3. Create your own calculator using keypad and lcd.

Interfaces

1. Buzzer beeping.
2. Buzzer and switch.
3. Ir Sensor.
4. LDR.
5. Relay
6. Motor clockwise
7. Motor anticlockwise.
8. Motor controlling using switches.

Timers and counters

1. Delay of 1 seconds using timers
2. Led blinking using timers
3. Counter value display seven segment or lcd.

Interrupt

1. **Timer interrupt.**
2. **Hardware interrupt.**

Serial communication(UART)

1. Simplex
2. Half Duplex
3. Full duplex
4. Serial interrupt

I2C

SPI

```
// PIC16F877A Configuration Bit Settings

// 'C' source line config statements

// CONFIG
#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT
enabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT
disabled)
#pragma config BOREN = OFF      // Brown-out Reset Enable bit (BOR
enabled)
#pragma config LVP = OFF        // Low-Voltage (Single-Supply) In-Circuit
Serial Programming Enable bit (RB3/PGM pin has PGM function;
low-voltage programming enabled)
#pragma config CPD = OFF        // Data EEPROM Memory Code
Protection bit (Data EEPROM code protection off)
#pragma config WRT = OFF        // Flash Program Memory Write Enable
bits (Write protection off; all program memory may be written to by EECON
control)
#pragma config CP = OFF         // Flash Program Memory Code Protection
bit (Code protection off)


#include <xc.h>
#define rs RC0
#define en RC2
#define _XTAL_FREQ 20000000
void cmd(unsigned char);
```

```
void display(unsigned char);
void init();
void cgram();
unsigned char
cust_char[]={0x17,0x14,0x14,0x1f,0x05,0x05,0x1d,0x00,0x0e,0x0a,0x1b,0
x11,0x11,0x11,0x1f,0x00};
void main(void) {
    unsigned char a[]="Hello world";
    TRISC=0;
    TRISB=0;
    init();
    cgram();
    display(0);
    display(1);
    while(1);
    return;
}
void cmd(unsigned char data){
   rs=0;
   PORTB=data & 0xf0;
   en=1;
   __delay_ms(100);
   en=0;
   PORTB=data << 4;
   en=1;
   __delay_ms(100);
   en=0;
}
void display(unsigned char data){
   rs=1;
   PORTB=data & 0xf0;
   en=1;
   __delay_ms(100);
   en=0;
   PORTB=data << 4;
```

```
   en=1;
   __delay_ms(100);
   en=0;
}
void init(){
    cmd(0x02);
    cmd(0x28);
    cmd(0x0e);
    cmd(0x06);
    cmd(0x80);
}
void cgram(){
    cmd(0x40);
    for(int i=0;i<16;i++){
        display(cust_char[i]);
    }
    cmd(0x80);
}
```

```
#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT enabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = OFF      // Brown-out Reset Enable bit (BOR enabled)
#pragma config LVP = OFF        // Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit (RB3/PGM pin has PGM function; low-voltage programming enabled)
#pragma config CPD = OFF        // Data EEPROM Memory Code Protection bit (Data EEPROM code protection off)
#pragma config WRT = OFF        // Flash Program Memory Write Enable bits (Write protection off; all program memory may be written to by EECON control)
#pragma config CP = OFF         // Flash Program Memory Code Protection bit (Code protection off)

#include<xc.h>



#define rs RD2
#define en RD3

#define R1 RB0
#define R2 RB1
#define R3 RB2
#define R4 RB3
#define C1 RB4
#define C2 RB5
#define C3 RB6
```

```
#define C4 RB7

void lcd_init();
void cmd(unsigned char a);
void dat(unsigned char b);
void show(unsigned char *s);
void lcd_delay();

unsigned char key();
void keyinit();

unsigned char
keypad[4][4]={{'7','8','9','/'},{'4','5','6','*'},{'1','2','3','-'},{'C','0','=','+'}};
unsigned char rowloc,colloc;

void main()
{
   unsigned int i;
   TRISD=0;
   lcd_init();
   keyinit();
   unsigned char b;
   cmd(0x80);
   show("  Enter the Key ");
   while(1)
   {
      cmd(0xc7);
      b=key();
      dat(b);

   }
}

void lcd_init()
{
```

```
    cmd(0x02);
    cmd(0x28);
    cmd(0x0e);
    cmd(0x06);
    cmd(0x80);
}

void cmd(unsigned char a)
{
    rs=0;
    PORTD&=0x0F;
    PORTD|=(a&0xf0);
    en=1;
    lcd_delay();
    en=0;
    lcd_delay();
    PORTD&=0x0f;
    PORTD|=(a<<4&0xf0);
    en=1;
    lcd_delay();
    en=0;
    lcd_delay();
}

void dat(unsigned char b)
{
    rs=1;
    PORTD&=0x0F;
    PORTD|=(b&0xf0);
    en=1;
    lcd_delay();
    en=0;
    lcd_delay();
    PORTD&=0x0f;
    PORTD|=(b<<4&0xf0);
```

```
    en=1;
    lcd_delay();
    en=0;
    lcd_delay();
}

void show(unsigned char *s)
{
    while(*s) {
        dat(*s++);
    }
}

void lcd_delay()
{
    unsigned int lcd_delay;
    for(lcd_delay=0;lcd_delay<=1000;lcd_delay++);
}

void keyinit()
{
    TRISB=0XF0;
    OPTION_REG&=0X7F;          //ENABLE PULL UP
}

unsigned char key()
{
    PORTB=0X00;
    while(C1&&C2&&C3&&C4);
    while(!C1||!C2||!C3||!C4) {
        R1=0;
        R2=R3=R4=1;
        if(!C1||!C2||!C3||!C4) {
            rowloc=0;
            break;
```

```
        }
        R2=0;R1=1;
        if(!C1||!C2||!C3||!C4) {
            rowloc=1;
            break;
        }
        R3=0;R2=1;
        if(!C1||!C2||!C3||!C4) {
            rowloc=2;
            break;
        }
        R4=0; R3=1;
        if(!C1||!C2||!C3||!C4){
            rowloc=3;
            break;
        }
    }
    if(C1==0&&C2!=0&&C3!=0&&C4!=0)
            colloc=0;
    else if(C1!=0&&C2==0&&C3!=0&&C4!=0)
            colloc=1;
    else if(C1!=0&&C2!=0&&C3==0&&C4!=0)
            colloc=2;
    else if(C1!=0&&C2!=0&&C3!=0&&C4==0)
            colloc=3;
    while(C1==0||C2==0||C3==0||C4==0);
    return (keypad[rowloc][colloc]);
}
```