1. ให้เชื่อมต่อสาย USB ของบอร์ดกับคอมพิวเตอร์ เปิดโปรแกรม Arduino จากนั้นทำการเขียนโปรแกรมที่ทำหน้าที่สั่งงานให้ LED Matrix 12x8 ที่อยู่บนบอร์ดไมโครคอนโทรลเลอร์แสดงรูปภาพตามค่าใน 2D array มีขนาดเป็นไบต์ที่มีจำนวนทั้งหมด 96 ไบต์ต่อ 1 fame ของภาพ โดยใช้ไลบรารีของ Arduino LED Matrix แล้วทำการ Upload โปรแกรมที่ได้ลงบนบอร์ด Arduino และให้ทดลองการทำงาน

```
#include "Arduino_LED_Matrix.h"          // Include the LED_Matrix Library
ArduinoLEDMatrix matrix;                 // Create an instance of the ArduinoLEDMatrix class

uint8_t frame[8][12] = {
  { 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0 },
  { 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0 },
  { 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1 },
  { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 },
  { 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1 },
  { 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1 },
  { 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0 },
  { 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0 }
};                                       // Pre-defined 2D array

void setup() {
  matrix.begin();                        // Initialize LED matrix
}

void smile(){
  frame[4][3] = 1;
  frame[4][8] = 1;
}

void face(){
  frame[4][3] = 0;
  frame[4][8] = 0;
}

void loop(){
face();
matrix.renderBitmap(frame, 8, 12);       // Display pattern on the LED matrix
delay(1000);

smile();
matrix.renderBitmap(frame, 8, 12);
delay(1000);
}
```

2. ให้แก้ไขโปรแกรมในข้อที่ 1 เพื่อให้ LED Matrix 12x8 แสดงผลเป็นรูปภาพอื่นตามแบบร่างที่กำหนดขึ้นเอง โดยใช้คำสั่งจากค่าใน 2D array ที่เขียนขึ้นใหม่ด้านล่าง

```
uint8_t frame[8][12] = {
  { 0 , 0 , 1 , 1 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 0 },
  { 0 , 1 , 1 , 1 , 1 , 0 , 1 , 1 , 1 , 1 , 0 , 0 },
  { 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 0 , 0 },
  { 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 0 , 0 },
  { 0 , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 0 , 0 , 0 },
  { 0 , 0 , 0 , 1 , 1 , 1 , 1 , 1 , 0 , 0 , 0 , 0 },
  { 0 , 0 , 0 , 0 , 1 , 1 , 1 , 0 , 0 , 0 , 0 , 0 },
  { 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 }};
```

หลักการของระบบตัวเลขจะถูกนำมาใช้ในการทำงานของสัญญาณดิจิตอล โดยแทนสภาวะการทำงานของสัญญาณ HIGH และ LOW ด้วยค่าคงที่เป็นเลขฐานสอง (Binary Number) ได้คือ 1 และ 0 โดยที่แต่ละสายข้อมูล 1 เส้น จะประกอบจากหน่วยข้อมูลที่เรียกว่า บิต (Bit) กลุ่มของตัวเลขฐานสองขนาด 8 บิตจะเรียกว่า ไบต์ (Byte) ใน 1 ไบต์จะมีจำนวนข้อมูลทั้งหมดเท่ากับ $2^8 = 256$ ค่า โดยไบต์ที่มีค่าเป็น 0 จะแสดงเป็น 00000000 และไบต์ที่ไม่ใช่ศูนย์จะเป็นการผสมกันของเลข 1 และ 0 เช่น 01001011 บิตที่อยู่ทางซ้ายสุดของชุดข้อมูลไบนารีเรียกว่า บิตที่มีนัยสำคัญมากที่สุด (Most Significant Bit) เรียกย่อว่า MSB และบิตขวาสุดจะเรียกว่า บิตที่มีนัยสำคัญน้อยที่สุด (Least Significant Bit) ย่อว่า LSB เพื่อให้เห็นค่าของข้อมูลแต่ละบิตในรูปเลขฐานสิบ จะเขียนให้อยู่ในรูปของเลขชี้กำลัง (Exponent) โดยให้ตำแหน่งบิตเป็นค่าของตัวเลขที่แสดงค่ายกกำลัง และมีฐานเป็นเลขสอง เริ่มจากบิตที่มีค่าน้อยที่สุดทางขวา (LSB) ฐานที่มีค่าเป็น 2 ยกกำลัง 0 จะมีค่าผลลัพธ์เท่ากับ 1 บิตต่อมาเรียงตามลำดับตำแหน่งบิตถัดไปทางด้านซ้ายที่ละ 1 บิต เลขชี้กำลังที่ได้แต่ละค่าจะเพิ่มขึ้นที่ละ 1 ดังนั้นหลักต่อมาจะได้ฐาน 2 ยกกำลัง 1 มีค่าเท่ากับ 2 เมื่อข้อมูลไบนารีมีขนาดเท่ากับ 1 ไบต์ บิตที่มีนัยสำคัญมากที่สุด (MSB) จะมีค่าเลขชี้กำลังเป็น 7 ถ้าให้ตำแหน่งบิตของข้อมูลเขียนย่อเป็นตัวอักษร D โดยเริ่มจาก D0, D1, D2 , D3, ฯลฯ ดังนั้นผลลัพธ์ของข้อมูลในแต่ละบิตในรูปเลขฐานสิบ (Decimal) จะได้ดังนี้

1 Byte

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Base^{exponent}	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal	128	64	32	16	8	4	2	1

การแสดงตัวเลขไบนารีเมื่อมีค่าของข้อมูลจำนวนมาก จะทำให้มีจำนวนหลักของข้อมูลที่มาก ซึ่งยุ่งยากต่อการเขียน เพื่อให้ง่ายเข้าโดยทั่วไปในการแสดงผลของข้อมูลจึงนิยมเขียนให้อยู่ในรูปของเลขฐานสิบหก (Hexadecimal) หรือย่อว่า HEX ดังนั้นเราจะได้จำนวนข้อมูลทั้งหมดเท่ากับ 16 ค่า คือ จาก 0 ถึง 15 การใช้เลขฐานสิบหกสามารถทำให้เขียนได้ง่ายขึ้น โดยแทน 0 ถึง 9 เหมือนเลขฐานสิบ และเฉพาะค่าเลข 10 ถึง 15 ซึ่งเป็นเลข 2 หลัก จะเปลี่ยนให้เป็น 1 หลัก โดยแทนด้วยตัวอักษร A ถึง F ตารางต่อไปนี้แสดงค่าของเลขฐานสิบหก (Hexadecimal) เทียบกับเลขฐานสอง (Binary) และเลขฐานสิบ (Decimal)

2 10 16

Binary	Decimal	Hexadecimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

Byte = 0-255 → ต่อ 1 จุด
Bit = 0-1

3. จากข้อที่ 1 เป็นการสร้างอาร์เรย์สองมิติเก็บค่าเป็นไบต์ ซึ่งวิธีนี้จะใช้หน่วยความจำมากกว่าที่จำเป็น เพราะว่า LED แต่ละตัวต้องการเพียงบิตเดียวในการจัดเก็บสถานะ ดังนั้นจึงให้ทำการเปลี่ยนขนาดของจุดภาพจาก 1 ไบต์ ไปเป็น 1 บิตต่อจุดภาพ ในการจัดเก็บภาพจึงเปลี่ยนมาใช้อาร์เรย์ของจำนวนเต็ม 32 บิต ทำให้วิธีนี้มีประสิทธิภาพในการใช้หน่วยความจำมากขึ้น จากตัวอย่างอาร์เรย์ข้อ 1 ด้านล่างให้แก้ไขรวมจำนวน 4 ค่าจากเลขฐาน 2 เปลี่ยนให้เป็นเลขฐาน 16 จำนวน 1 ค่า

12 (จำนวน) ฐาน 16 = 24 ตัว

8 (จำนวน)

{ 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0 } แปลงเป็นเลขฐาน 16 เท่ากับ 3 , F , C

{ 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0 } แปลงเป็นเลขฐาน 16 เท่ากับ 6 , 0 , 6

{ 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1 } แปลงเป็นเลขฐาน 16 เท่ากับ D , 9 ; B

{ 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 } แปลงเป็นเลขฐาน 16 เท่ากับ 8 , 0 , 1

{ 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1 } แปลงเป็นเลขฐาน 16 เท่ากับ 9 , 0 , 9

{ 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1 } แปลงเป็นเลขฐาน 16 เท่ากับ C ; F , 3

{ 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0 } แปลงเป็นเลขฐาน 16 เท่ากับ 6 , 0 , 6

{ 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0 } แปลงเป็นเลขฐาน 16 เท่ากับ 3 , F , C

เมื่อเขียนเป็นเลขฐาน 16 ขนาด 32 บิตจะได้ 3FC606D9 , B801909C , F36063FC

4. จากค่าในข้อที่ 3 เมื่อนำไปสร้างอาร์เรย์ใหม่ที่เป็นเลขจำนวนเต็ม 32 บิตแทนในข้อที่ 1 โดยที่ LED Matrix จะมีจุดภาพ 12 x 8 = 96 ดังนั้นอาร์เรย์จะมีค่าที่เก็บทั้งหมด 96/32 = 3 ค่า โดยโปรแกรมที่แก้ไขแล้วเพื่อแสดงผลใน LED Matrix 12x8 เป็นดังนี้

```
#include "Arduino_LED_Matrix.h"
ArduinoLEDMatrix matrix;

void setup() {
  matrix.begin();
}

const uint32_t smile[] = {
    0x3fc606d9,
    0xb801909c,
    0xf36063fc
};
const uint32_t happy[] = {
    0x19819,
    0x80000001,
    0x81f8000
};

void loop(){
  matrix.loadFrame(smile);
  delay(500);

  matrix.loadFrame(happy);
  delay(500);
}
```

5. ให้แก้ไขโปรแกรมในข้อที่ 4 เพื่อให้ LED Matrix 12x8 แสดงผลเป็นรูปภาพอื่น

6. ให้เขียนโปรแกรมเพื่อให้ LED Matrix 12x8 แสดงผลกราฟิกชื่อนักศึกษาเป็นภาษาไทยเคลื่อนไหวจากด้านขวาไปซ้าย โดยมีรูปแบบการเขียนโปรแกรมตามตัวอย่างด้านล่าง

```
#include "Arduino_LED_Matrix.h"          // Include the LED_Matrix library
ArduinoLEDMatrix matrix;                 // Create an instance of the ArduinoLEDMatrix class

const uint32_t frames[][4] = {
        { 0x0, 0x10010010, 0x1000000, 100 },
        { 0x100, 0x20030020, 0x2000000, 100 },
        { 0x300, 0x40070040, 0x4000000, 100 },
        { 0x600, 0x900f0090, 0x9000000, 100 },
        { 0xc01, 0x201e0120, 0x12000000, 100 },
        { 0x1902, 0x503d0250, 0x25000000, 100 },
        { 0x3304, 0xa07b04a0, 0x4a000000, 100 },
        { 0x6709, 0x40f70940, 0x94000000, 100 },
        { 0xce12, 0x91ee1291, 0x29000000, 100 },
        { 0x19c25, 0x23bc2522, 0x52000000, 100 },
        { 0x3394a, 0x57b94a54, 0xa5000000, 100 },
        { 0x67394, 0xaf7294a9, 0x4b000000, 100 },
        { 0xce729, 0x4ee42942, 0x97000000, 100 },
        { 0x9ce52, 0x9dc95295, 0x2e000000, 100 },
        { 0x39ca5, 0x2b92a52a, 0x5c000000, 100 },
        { 0x7394a, 0x57254a54, 0xb8000000, 100 },
        { 0xe7294, 0xae4a94a9, 0x71000000, 100 },
        { 0xce429, 0x4c942942, 0xe3000000, 100 },
        { 0x9c952, 0x99295295, 0xc6000000, 100 },
        { 0x392a5, 0x2252a52b, 0x8c000000, 100 },
        { 0x7254a, 0x44a44a47, 0x19000000, 100 },
        { 0xe4b94, 0x9949949e, 0x33000000, 100 },
        { 0xc9729, 0x2292292c, 0x67000000, 100 },
        { 0x92e52, 0x45245248, 0xce000000, 100 },
        { 0x25da4, 0x9a49a491, 0x9d000000, 100 },
        { 0x4ba49, 0x34924923, 0x3a000000, 100 },
        { 0x97492, 0x69259246, 0x74000000, 100 },
        { 0x2e924, 0xd24b249c, 0xe9000000, 100 },
        { 0x5d249, 0xa4964929, 0xd2000000, 100 },
        { 0xba493, 0x592d9253, 0xa4000000, 100 },
        { 0x74926, 0xa25a24a7, 0x49000000, 100 },
        { 0xe934d, 0x44b4494e, 0x93000000, 100 },
        { 0xd269a, 0x9969929d, 0x26000000, 100 },
        { 0xa4c35, 0x22d2252a, 0x4c000000, 100 },
        { 0x4986a, 0x45a44a44, 0x98000000, 100 },
        { 0x930d4, 0x8b489489, 0x30000000, 100 },
        { 0x260a9, 0x6902902, 0x60000000, 100 },
        { 0x4c052, 0xd205204, 0xc0000000, 100 },
        { 0x980a4, 0xa40a409, 0x80000000, 100 },
        { 0x30048, 0x4804803, 0x0, 100 },
        { 0x60090, 0x9009006, 0x0, 100 },
        { 0xc0020, 0x200200c, 0x0, 100 },
        { 0x80040, 0x4004008, 0x0, 100 },
        { 0x80, 0x8008000, 0x0, 100 },
        { 0x0, 0x0, 0x0, 100 }
};

void setup() {
        matrix.loadSequence(frames);
        matrix.begin();
        matrix.play(true);
}

void loop() {
}
```