```
= Name        : Pong.c                                                    =

#define F_CPU 8000000L // 8 MHz
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>



// ============================================================================
// ====         VARIABLES               ====================================
// ============================================================================
// Global variables declered
//              Name                    Description                          interval
//-----------------------------------------------------------------------------------------
unsigned char window_length = 128;  // Width of the screen
unsigned char window_height = 64;   // Height of the screen

unsigned char ball_x;               // The balls upper left hand side corners x coordinate
unsigned char ball_y;               // The balls upper left hand side corners y coordinate
unsigned char ball_length;          // The balls side length
unsigned char ball_speed;           // Number of pixels the ball move each tick (0-5)
unsigned char ball_direction;       // Direction of the balls movement          (0-5)

unsigned char score_p1;             // Score of player 1
unsigned char score_p2;             // Score of player 2

unsigned char paddle1_x;            // The left paddles upper left hand side corners x
coordinate
unsigned char paddle1_y;            // The left paddles upper left hand side corners y
coordinate

unsigned char paddle2_x;            // The right paddles upper left hand side corners y
coordinate
unsigned char paddle2_y;            // The right paddles upper left hand side corners y
coordinate

unsigned char paddle_length;        // The height of the paddle in pixels       (1-10)
unsigned char paddle_speed;         // Unused right now
unsigned char computer_speed;       // Number of pixels the computer can move each tick
unsigned char nbrPlayers;           // Number of players to start the next game with
unsigned char activeGame;           // 0: Menu, 1: running game, 2: game finished
unsigned char tempNBR;              // a gloabal temp variable
unsigned char game_speed;           // Delay between each tick                  (0-255)


// ============================================================================
// ====         ADC FUNCTIONS              ====================================
// ============================================================================
// Functions for ADC-read found on AVR freaks

// initialize adc
void adc_init()
{
    // AREF = AVcc
    ADMUX = (0<<REFS0|0<<REFS1);

    // ADC Enable and prescaler of 128
    // 16000000/128 = 125000
    ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
}
```

```
// read adc value
uint16_t adc_read(uint8_t ch)
{
    // select the corresponding channel 0~7
    // ANDing with '7' will always keep the value
    // of 'ch' between 0 and 7
    ch &= 0b00000111;  // AND operation with 7
    ADMUX = (ADMUX & 0xF8)|ch;     // clears the bottom 3 bits before ORing

    // start single conversion
    // write '1' to ADSC
    ADCSRA |= (1<<ADSC);

    // wait for conversion to complete
    // ADSC becomes '0' again
    // till then, run loop continuously
    while(ADCSRA & (1<<ADSC));

    return (ADC);
}

// ================================================================================
// ====         BUTTON AND INTERRUPT FUNCTIONS              =================
// ================================================================================
// Functions to catch the buttons interrupt

// Function for players 2s button
ISR(INT0_vect) // Knapp P1
{
    switch (activeGame){
        case 0:
            nbrPlayers = tempNBR;
        break;
        case 1:
         if( (DDRC|0b10111111) == 0xFF){
            DDRC &= 0b10111111;
            //MAYHEM FUNCTION!!
         }
        break;
        case 2:
         activeGame = 0;
        break;
        case 3:
         activeGame = 0;
        break;
    }
}

// Function for players 2s button
ISR(INT1_vect)
{
    switch (activeGame){
        case 0:
            nbrPlayers = tempNBR;
        break;
        case 1:
         if( (DDRC|0b01111111) == 0xFF){
            DDRC &= 0b01111111;
            //MAYHEM FUNCTION!!
         }
```

```
        break;
        case 2:
         activeGame = 0;
        break;
        case 3:
         activeGame = 0;
        break;
    }
}

// ================================================================================
// ====         STARTUP FUNCTIONS                                 ================
// ================================================================================

// Sets start values for ports and data direction
void setStartValues(){

// Enable the interrupt on INT0 and INT1
 cli();
 GICR = 0b11000000;
 MCUCR = 0b00001111;
 sei();

//Sets the data direction at start
 DDRA = 0x00;
 DDRB = 0b11111111;
 DDRC &= 0b00111111;
 DDRD  = 0b11110011;

//Sets the inital values for ports
 PORTD = _BV(PD4);
 //PORTD |= _BV(PD7);
}

// ================================================================================
// ====         MENU FUNCTIONS              ================================
// ================================================================================

// Joystick read for the menu
unsigned char changeNbrPlayers(unsigned char temp){
    unsigned int ADC_value = adc_read(0);                          //RANGE: 1023 - 0

    if(ADC_value > 610 || ADC_value < 490){
        if(temp==2){
          return 1;
        }else{
         return 2;
        }
    }
    return temp;
}

// Main menu
void menu(){
    activeGame=3;
    // Shows the startup splashscreen
    TurnOnLCD();
    startAnimation();

    // Select number of players
    tempNBR = 2;
```

```
    clearDisplay();
    drawMenu();
    TurnOnLCD();

    //Sets menu variables
    activeGame = 0;
    nbrPlayers = 0;

    // Menu loop for player selection
    do{
        tempNBR = changeNbrPlayers(tempNBR);
        writeNbrPlayers(tempNBR);
        _delay_ms(300);
    }while(nbrPlayers==0);

    // Starts the game
    switch (nbrPlayers){
        case 1:
         playGame1P();
        break;
        case 2:
         playGame2P();
        break;
    }
}

// Updates the number of players screen based on the input
void writeNbrPlayers(unsigned char nbr){
    clearDisplay();
    switch (nbr){
     case 1:
        drawMenu();
        write_1(58,35);

     break;
     case 2:
        drawMenu();
        write_2(58,35);
     break;
    }
}


// ============================================================================
// ====         MOVE FUNCTIONS              ================================
// ============================================================================
/**  --- MOVEBALL ---
  * Function to processes the events from each cycle in the game
  *       Directions    **      The balls direction of travel
  *     5   \   /   0    *
  *     4 <--   --> 1    *
  *     3   /   \   2    *
  **                     **
  *
  */

//Function to move the ball
unsigned char moveBall(){
    if(ball_direction > 2){
     ball_x -= ball_speed;
    } else{
```

```
 ball_x += ball_speed;
}

if(ball_direction == 2 || ball_direction == 3){
 if(ball_y < window_height-ball_length-ball_speed){
  ball_y += ball_speed;
 }else{
    ball_y = window_height-ball_length-1;
    switch(ball_direction){
     case 2:
        ball_direction = 0;
     break;
     case 3:
        ball_direction = 5;
     break;
    }
 }
}else if(ball_direction == 5 || ball_direction == 0){
 if(ball_y > ball_speed){
  ball_y -= ball_speed;
 }else{
    ball_y = 0;
    switch(ball_direction){
     case 0:
        ball_direction = 2;
     break;
     case 5:
        ball_direction = 3;
     break;
    }
 }
}

//Collision with right paddle
if(ball_x + ball_length >= paddle2_x){
 unsigned char hit = 0;
 for(unsigned char k = paddle2_y; k <= paddle2_y+paddle_length; k++){
    for(unsigned char j = ball_y;j < ball_y + ball_length ;j++){
        if(j == k){
            hit = 1;
            break;
        }
    }
    if(hit == 1){
        break;
    }
 }

 // If collision has occured change the direction
 if(hit==1){
    if( (paddle2_y-ball_y) < 2){
        ball_direction = 3;
    }else if((paddle2_y-ball_y) < 6){
        ball_direction = 4;
    }else{
        ball_direction = 5;
    }

    // If the ball bounch move ball away from paddle
    if(ball_direction > 2){
     ball_x -= ball_speed;
```

```
    } else{
     ball_x += ball_speed;
    }

    // If the ball bounch move ball away from paddle
    if(ball_direction == 0 || ball_direction == 5){
     ball_y += ball_speed;
    }else if(ball_direction == 4 || ball_direction == 1){
     //nbsp;
    }else{
     ball_y -= ball_speed;
    }

 }else{
    updateP1Score();
    return 1;
 }
}

// Collision with left paddle
if(ball_x <= paddle1_x){
 unsigned char hit = 0;
 for(unsigned char k = paddle1_y; k <= paddle1_y+paddle_length; k++){
    for(unsigned char j = ball_y;j<ball_y + ball_length;j++){
        if(j == k){
            hit = 1;
            break;
        }
    }
    if(hit == 1){
        break;
    }
 }
 // If collision has occured change the direction
 if(hit==1){
    if( (paddle1_y-ball_y) < 2){
        ball_direction = 2;
    }else if((paddle1_y-ball_y) < 6){
        ball_direction = 1;
    }else{
        ball_direction = 0;
    }

    // If the ball bounch move ball away from paddle
    if(ball_direction > 2){
     ball_x -= ball_speed;
    } else{
     ball_x += ball_speed;
    }

    if(ball_direction == 0 || ball_direction == 5){
     ball_y += ball_speed;
    }else if(ball_direction == 4 || ball_direction == 1){
     //nbsp;
    }else{
     ball_y -= ball_speed;
    }
 }else{
    updateP2Score();
    return 1;
 }
```

```
    }
    return 0;
}

// Function to move the right paddle in a one player game
void computerMovePaddle(){
    if(ball_y > (paddle2_y + paddle_length)){
        paddle2_y += computer_speed;
        if((paddle2_y + paddle_length) > (window_height)){
            paddle2_y = window_height - paddle_length - 1;
        }
    }else{
        if(paddle2_y - computer_speed < 0){
         paddle2_y = 0;
        }else{
         paddle2_y -= computer_speed;
        }
    }
}


// Moves the left paddle
void moveLeftPaddle(){
    unsigned int ADC_value = adc_read(0);// Reads the PA0 value (Joystick 1)

    // Switches the ADC value to a movement of 0-3 pixels up or down
    if(ADC_value > 608) // Move down
    {
     ADC_value = ((ADC_value-608)/138)+1;
     if((window_height - 1 - paddle1_y - paddle_length) <= ADC_value){
        paddle1_y = window_height - paddle_length - 1;
     }else{
        paddle1_y += ADC_value;
     }
    }else if(ADC_value<510){
      ADC_value = 3 - ((ADC_value/173));
      if( paddle1_y > ADC_value ){
        paddle1_y -= ADC_value;
      }else{
        paddle1_y = 0;
      }
    }
}

// Moves the right paddle
void moveRightPaddle(){
    unsigned int ADC_value = adc_read(1); // Reads the PA1 value (Joystick 2)

    // Switches the ADC value to a movement of 0-3 pixels up or down
    if(ADC_value > 608)
    {
     ADC_value = ((ADC_value-608)/138) + 1;
     if((window_height - 1 - paddle2_y - paddle_length) < ADC_value)
     {
        paddle2_y = window_height - paddle_length - 1;
     }else{
        paddle2_y += ADC_value;
     }
    }else if(ADC_value<510){
      ADC_value = 3-((ADC_value/173));
      if( paddle2_y > ADC_value )
```

```
      {
        paddle2_y -= ADC_value;
      }else{
        paddle2_y = 0;
      }
    }
}

// ================================================================================
// ====          GAME FUNCTIONS              ==================================
// ================================================================================

//Initiate a two player game
void playGame2P(){
 initGame(); // Initiate a new game
 newSet();   //Starts a new set
 updateScreen(); // Updates the screen

 //Main loop for a two player game
 while(score_p1 < 3 && score_p2 < 3){
    moveLeftPaddle();
    moveRightPaddle();
    if(moveBall()){ // If moveBall != 0 a goal has been scored
        goalAnimation();
        newSet();
    }
    updateScreen(); //Updates the screen after all the ticks movements
    _delay_ms(game_speed);  //Delay to get the right speed of the game
 }

  clearDisplay();
  splashScreen();
  activeGame = 2;
  while(activeGame == 2);

}

// Initiate a one player game
void playGame1P(){
 initGame();    //Initiate a new game
 newSet();      //Starts a new set
 updateScreen();//Updates the screen
 computer_speed = 1; //Set the number of pixel the computer paddle can move each tick

 // Main loop for a one player game
 while(score_p1 < 3 && score_p2 < 3){
    moveLeftPaddle();
    computerMovePaddle();
    if(moveBall()){ // If moveball != 0 a goal has been scored
        goalAnimation();
        newSet();
    }
    updateScreen(); //Updates the screen after all the ticks movements
    _delay_ms(game_speed);  //Delay to get the right speed of the game
 }
  activeGame = 2;
  while(activeGame == 2);
}

// Sets start values for a new match
void initGame(){
```

```
 ball_length = 3;
 ball_speed = 2;
 paddle_length = 11;
 activeGame = 1;
 game_speed = 40;       //0-255 ms delay for each tick
 score_p1 = 0;
 score_p2 = 0;

 scoreResetLED();
}

// Sets startvalues for a new set
void newSet(){
    paddle1_x = 1;
    paddle1_y = window_height / 2;

    paddle2_x = window_length - 2;
    paddle2_y = window_height / 2;

    ball_direction = rand() % 6;

    ball_x = window_length/2;
    ball_y = window_height/2;

    DDRC |= 0b11000000;
}



// ============================================================================
// ====         GRAPHIC FUNCTIONS           ================================
// ============================================================================

// Updates the screen each tick with the current position of all the objects
void updateScreen(){
    clearDisplay();
    drawP1Paddle();
    drawP2Paddle();
    drawBall();
    //TurnOnLCD();
}

// Clears the entire screen
void clearDisplay()
{
 for(unsigned char x = 0; x < 8; x++)
 {
     //Set x address
     // RS RW D7 D6 D5 D4 D3 D2 D1 D0
     // 0  0  1  0  1  1  1 [ PAGE ]
     PORTB =  0b10111000 + x; // Page select 1 0 1 1 1 0 0 0
     PORTD &= 0b10011111; //Sets RS and RW to 0
     PORTD |= 0b00010011;

     LCD_E_Toggle();
  for(unsigned char y = 0; y < 64; y++)
  {
     //Clear data
     PORTD &= 0b11011111; // Sets RW to 0
     PORTD |= 0b01010011; //Sets RS to 1
     PORTB =  0b00000000;
```

```
     LCD_E_Toggle();
  }
 }
}

// Draws the left paddle (Player 1)
void drawP1Paddle(){
 for(unsigned char k = paddle1_y; k <= paddle1_y + paddle_length; k++){
    setPixelOn(paddle1_x, k);
    setPixelOn(paddle1_x-1,k);
 }
}

// Draws the right paddle (Player 2)
void drawP2Paddle(){
 for(unsigned char k = paddle2_y; k <= paddle2_y + paddle_length; k++){
    setPixelOn(paddle2_x, k);
    setPixelOn(paddle2_x+1,k);
 }
}

// Draws the ball on the screen
void drawBall(){
 for(unsigned char k = ball_x; k < ball_x + ball_length; k++){
  for(unsigned char i = ball_y; i < ball_y + ball_length; i++){
    setPixelOn(k,i);
  }
 }
}

// Draws the text "ANTAL SPELARE" on the screen
void drawMenu(){
    write_A(0,9);
    write_N(14,9);
    write_T(24,9);
    write_A(33,9);
    write_L(47,9);

    write_S(61,9);
    write_P(71,9);
    write_E(80,9);
    write_L(88,9);
    write_A(97,9);
    write_R(111,9);
    write_E(120,9);

    //write_1(38,35);
    //write_2(71,35);
}

// Draws the screen showed after each goal
void goalAnimation(){
    clearDisplay();
    write_M(38,20);
    write_A(55,20);
    write_A_ring(55,20);
    write_L(73,20);
    write_exclamation(84,20);
    LED_Tripple_Blink();
    setScoreLED();
```

```
}

// Draws the startup splash screen and LED flashes
void startAnimation(){
 //LED_InOut();
 //LED_RandomFuck1();
 //LED_Tripple_Blink();
 clearDisplay();
 splashScreen();
 while(activeGame==3){
  LED_SideToSide();
 }
}

// ============================================================================
// ====          LCD FUNCTIONS               ================================
// ============================================================================

//Function the turn on a specific pixel at x,y
void setPixelOn(unsigned char x, unsigned char y){

 // Sets which of the two display chips is responsible for this pixel
 unsigned char disp;
 if(x>63){
  x -= 64;
  disp = 0b00000001;
 }else{
  disp = 0b00000010;
 }

 // translates the x,y to the corisponding x,y of the screen
 unsigned char x_cord = 63-y;
 unsigned char y_cord = 63-x;

 //Set vertical value! (Screens x)
 // RS RW D7 D6 D5 D4 D3 D2 D1 D0
 // 0  0  1  0  1  1  1 [ PAGE ]
 PORTB =  0b10111000 + (x_cord/8); // Page select 1 0 1 1 1 0 0 0
 PORTD &= 0b10011100;              //Sets RS and RW to 0
 PORTD |= 0b00010000 + disp;

 LCD_E_Toggle();

 //Set Horizontal address (Screens y)
 // RS RW D7 D6 D5 D4 D3 D2 D1 D0
 // 0  0  0  1 [    address    ];
 PORTB =  0b01000000 + y_cord;  // y address
 PORTD &= 0b10011100;           //Sets RS and RW to 0
 PORTD |= 0b00010000 + disp;

 LCD_E_Toggle();

 // Read current status the byte
 DDRB   = 0b00000000;
 PORTD &= 0b11111100;
 PORTD |= 0b01100000 + disp; // Sets RS and RW to 1 for read
 //E Toggle /wo ready
 PORTD |= 0b10000000;   // E ==> Hög
 PORTD &= 0b01111111;   // E ==> Låg
 PORTD |= 0b10000000;   // E ==> Hög
 LCD_Ready();
```

```
 unsigned char temp = PINB;
 PORTD &= 0b01111111;   // E ==> Låg

 DDRB   = 0b11111111;
 LCD_Ready();

 //Set horizontal value (Screens y)
 // RS RW D7 D6 D5 D4 D3 D2 D1 D0
 // 0  0  0  1 [    address     ]
 PORTB =  0b01000000 + y_cord;  // y address
 PORTD &= 0b10011100;           //Sets RS and RW to 0
 PORTD |= 0b00010000 + disp;

 LCD_E_Toggle();

 // Write data
 PORTD &= 0b11011100;           // Sets RW to 0
 PORTD |= 0b01000000 + disp;    // Sets RS to 1
 PORTB = temp;                  // PORTB gets the bytes current value
 PORTB |= (1 <<(x_cord%8));     //Add the bit that should be turn on ;)

 LCD_E_Toggle();
}

//Function the turn off a specific pixel at x,y
void setPixelOff(unsigned char x, unsigned char y){

 // Sets which of the two display chips is responsible for this pixel
 unsigned char disp;
 if(x>63){
  x -= 64;
  disp = 0b00000001;
 }else{
  disp = 0b00000010;
 }

 unsigned char x_cord = 63-y;
 unsigned char y_cord = 63-x;

 //Set vertical value! (Screens x)
 // RS RW D7 D6 D5 D4 D3 D2 D1 D0
 // 0  0  1  0  1  1  1 [ PAGE ]
 PORTB =  0b10111000 + (x_cord/8); // Page select 1 0 1 1 1 0 0 0
 PORTD &= 0b10011100;              //Sets RS and RW to 0
 PORTD |= 0b00010000 + disp;

 LCD_E_Toggle();

 //Set Horizontal address (Screens y)
 // RS RW D7 D6 D5 D4 D3 D2 D1 D0
 // 0  0  0  1 [    address    ];
 PORTB =  0b01000000 + y_cord;  // y address
 PORTD &= 0b10011100;           //Sets RS and RW to 0
 PORTD |= 0b00010000 + disp;

 LCD_E_Toggle();

 // Read current status the byte
 DDRB   = 0b00000000;
 PORTD &= 0b11111100;
 PORTD |= 0b01100000 + disp; // Sets RS and RW to 1 for read
```

```
 //E Toggle /wo ready
 PORTD |= 0b10000000;   // E ==> Hög
 PORTD &= 0b01111111;   // E ==> Låg
 PORTD |= 0b10000000;   // E ==> Hög
 LCD_Ready();
 unsigned char temp = PINB;
 PORTD &= 0b01111111;   // E ==> Låg

 DDRB   = 0b11111111;
 LCD_Ready();

 //Set horizontal value (Screens y)
 // RS RW D7 D6 D5 D4 D3 D2 D1 D0
 // 0  0  0  1 [    address    ];
 PORTB =  0b01000000 + y_cord;  // y address
 PORTD &= 0b10011100;           //Sets RS and RW to 0
 PORTD |= 0b00010000 + disp;

 LCD_E_Toggle();

 // Write data
 PORTD &= 0b11011100;           // Sets RW to 0
 PORTD |= 0b01000000 + disp;    // Sets RS to 1
 PORTB = temp;                  // PORTB gets the bytes current value
 PORTB &= (0 <<(x_cord%8));     //Add the bit that should be turn on ;)

 LCD_E_Toggle();
}


//Function to set the E signal from high to low
void LCD_E_Toggle(){
 PORTD |= 0b10000000;   // E ==> Hög
 PORTD &= 0b01111111;   // E ==> Låg

 LCD_Ready();
}

// A small delay function to give the screen time to process the instruction
void LCD_Ready(){
 for(unsigned char k=0; k<10;k++){
  //asm volatile ("nop");
 }
}

//Turns on the screen
void TurnOnLCD()
{
 PORTB = 0b00111111;
 PORTD = 0b00010011;

 LCD_E_Toggle();
}

//Turns off the screen
void TurnOffLCD()
{
 PORTB = 0b00111110;
 PORTD = 0b00010011;

 LCD_E_Toggle();
```

```
}


// ============================================================================
// ====          LED FUNCTIONS                ==================================
// ============================================================================

// Updates the score for player 1 at a goal
void updateP1Score(){
    score_p1++;
    setScoreLED();
}

//Upadets the score for player 2 at a goal
void updateP2Score(){
    score_p2++;
    setScoreLED();
}

// Turns off all the white LEDs
void scoreResetLED(){
    DDRA = DDRA & 0x03; //00000011
}

// Function to controll the red LEDs
void mayhemLED(unsigned short int LED, unsigned short int func){
 switch (LED){
    case 1:
        switch (func){
            case 0:
                PORTA = PORTA & 0xFE;   //11111110
            break;
            case 1:
                PORTA = PORTA | 0x01;   //00000001
            break;
        }
    break;
    case 2:
        switch(func){
            case 0:
                PORTA = PORTA & 0xFD;   //11111101
            break;
            case 1:
                PORTA = PORTA | 0x02;   //00000010
            break;
        }
    break;
 }
}

// Set the white LEDs to reflect the current score
void setScoreLED(){
    DDRA &= 0b00000011;
    switch (score_p1){
     case (1):
        DDRA |= 0b00000100;
     break;
     case (2):
        DDRA |= 0b00001100;
     break;
     case (3):
```

```
        DDRA |= 0b00011100;
     break;
    }

    switch (score_p2){
     case (1):
        DDRA |= 0b00100000;
     break;
     case (2):
        DDRA |= 0b01100000;
     break;
     case (3):
        DDRA |= 0b11100000;
     break;
    }
}

// ============================================================================
// ====         MAIN                          ================================
// ============================================================================

// The main loop for the game
void main(void) {
 setStartValues(); // Startup values
 adc_init();       // Turns on the ADC
 while(1){         // Infintie loop to cycle back to the menu
  menu();
 }
}


// ============================================================================
// ====         LED LIGHT SHOWS             ================================
// ============================================================================

void LED_Tripple_Blink(){
 DDRA = 0b11111100;
 _delay_ms(200);
 DDRA = 0b00000000;
 _delay_ms(200);
 DDRA = 0b11111100;
 _delay_ms(200);
 DDRA = 0b00000000;
 _delay_ms(200);
 DDRA = 0b11111100;
 _delay_ms(200);
 DDRA = 0b00000000;
 //_delay_ms(100);
}

void LED_SideToSide(){
 DDRC &= 0b01111111;
 DDRC |= 0b01000000;
 _delay_ms(200);
 DDRC &= 0b00111111;

 DDRA = 0b00000100;
 _delay_ms(200);
 DDRA = 0b00001000;
 _delay_ms(200);
 DDRA = 0b00010000;
```

```
 _delay_ms(200);

 DDRA = 0b10000000;
 _delay_ms(200);
 DDRA = 0b01000000;
 _delay_ms(200);
 DDRA = 0b00100000;
 _delay_ms(200);

 DDRA = 0b00000000;
 //_delay_ms(200);

 DDRC &= 0b10111111;
 DDRC |= 0b10000000;
 _delay_ms(200);
 DDRC &= 0b00111111;
}

/*
void LED_InOut(){
 DDRA = 0b11111100;
 _delay_ms(250);
 DDRA = 0b10000100;
 _delay_ms(250);
 DDRA = 0b01001000;
 _delay_ms(250);
 DDRA = 0b00110000;
 _delay_ms(250);
 DDRA = 0b01001000;
 _delay_ms(250);
 DDRA = 0b10000100;
 _delay_ms(250);
}



void LED_RandomFuck1(){

 DDRA = 0b00000100;
 _delay_ms(250);
 DDRA = 0b00001100;
 _delay_ms(250);
 DDRA = 0b00011100;
 _delay_ms(250);

 DDRA = 0b10011100;
 _delay_ms(250);
 DDRA = 0b010111100;
 _delay_ms(250);
 DDRA = 0b111111100;
 _delay_ms(250);

 DDRA = 0b11111000;
 _delay_ms(250);
 DDRA = 0b11110000;
 _delay_ms(250);
 DDRA = 0b11100000;
 _delay_ms(250);

 DDRA = 0b01100000;
 _delay_ms(250);
```

```
 DDRA = 0b00100000;
 _delay_ms(250);
 DDRA = 0b00000000;
 _delay_ms(250);
}
/**/


// ================================================================================
// ====         Letters                        ================================
// ================================================================================

//All letters have height 14 px
//Letter spacing 3 px (except T followed by A -> 2px)

// Writes an A with the upper left hand side corner at x,y
//Width 13 px
void write_A(unsigned char x,unsigned char y){
    setPixelOn(x,y+13);
    setPixelOn(x,y+12);

    setPixelOn(x+1,y+11);
    setPixelOn(x+1,y+10);

    setPixelOn(x+2,y+9);
    setPixelOn(x+2,y+8);

    setPixelOn(x+3,y+8);
    setPixelOn(x+3,y+7);
    setPixelOn(x+3,y+6);

    setPixelOn(x+4,y+8);
    setPixelOn(x+4,y+5);
    setPixelOn(x+4,y+4);

    setPixelOn(x+5,y+8);
    setPixelOn(x+5,y+3);
    setPixelOn(x+5,y+2);

    setPixelOn(x+6,y+8);
    setPixelOn(x+6,y+1);
    setPixelOn(x+6,y+0);

    setPixelOn(x+7,y+8);
    setPixelOn(x+7,y+3);
    setPixelOn(x+7,y+2);

    setPixelOn(x+8,y+8);
    setPixelOn(x+8,y+5);
    setPixelOn(x+8,y+4);

    setPixelOn(x+9,y+8);
    setPixelOn(x+9,y+7);
    setPixelOn(x+9,y+6);

    setPixelOn(x+10,y+9);
    setPixelOn(x+10,y+8);

    setPixelOn(x+11,y+11);
    setPixelOn(x+11,y+10);
```

```
    setPixelOn(x+12,y+13);
    setPixelOn(x+12,y+12);
}

// Writes an N with the upper left hand side corner at x,y
//width 9 px
void write_N(unsigned char x,unsigned char y){
    setPixelOn(x,y+13);
    setPixelOn(x,y+12);
    setPixelOn(x,y+11);
    setPixelOn(x,y+10);
    setPixelOn(x,y+9);
    setPixelOn(x,y+8);
    setPixelOn(x,y+7);
    setPixelOn(x,y+6);
    setPixelOn(x,y+5);
    setPixelOn(x,y+4);
    setPixelOn(x,y+3);
    setPixelOn(x,y+2);
    setPixelOn(x,y+1);
    setPixelOn(x,y);

    setPixelOn(x+1,y);
    setPixelOn(x+1,y+1);

    setPixelOn(x+2,y+2);
    setPixelOn(x+2,y+3);

    setPixelOn(x+3,y+4);
    setPixelOn(x+3,y+5);

    setPixelOn(x+4,y+6);
    setPixelOn(x+4,y+7);

    setPixelOn(x+5,y+8);
    setPixelOn(x+5,y+9);

    setPixelOn(x+6,y+10);
    setPixelOn(x+6,y+11);

    setPixelOn(x+7,y+12);
    setPixelOn(x+7,y+13);

    setPixelOn(x+8,y+13);
    setPixelOn(x+8,y+12);
    setPixelOn(x+8,y+11);
    setPixelOn(x+8,y+10);
    setPixelOn(x+8,y+9);
    setPixelOn(x+8,y+8);
    setPixelOn(x+8,y+7);
    setPixelOn(x+8,y+6);
    setPixelOn(x+8,y+5);
    setPixelOn(x+8,y+4);
    setPixelOn(x+8,y+3);
    setPixelOn(x+8,y+2);
    setPixelOn(x+8,y+1);
    setPixelOn(x+8,y);
}

// Writes an T with the upper left hand side corner at x,y
//width 9 px
```

```
void write_T(unsigned char x,unsigned char y){
    setPixelOn(x,y);
    setPixelOn(x+1,y);
    setPixelOn(x+2,y);
    setPixelOn(x+3,y);
    setPixelOn(x+4,y);
    setPixelOn(x+5,y);
    setPixelOn(x+6,y);
    setPixelOn(x+7,y);
    setPixelOn(x+8,y);

    setPixelOn(x+4,y+1);
    setPixelOn(x+4,y+2);
    setPixelOn(x+4,y+3);
    setPixelOn(x+4,y+4);
    setPixelOn(x+4,y+5);
    setPixelOn(x+4,y+6);
    setPixelOn(x+4,y+7);
    setPixelOn(x+4,y+8);
    setPixelOn(x+4,y+9);
    setPixelOn(x+4,y+10);
    setPixelOn(x+4,y+11);
    setPixelOn(x+4,y+12);
    setPixelOn(x+4,y+13);
}

// Writes an L with the upper left hand side corner at x,y
//width 8 px
void write_L(unsigned char x,unsigned char y){
    setPixelOn(x,y+1);
    setPixelOn(x,y+2);
    setPixelOn(x,y+3);
    setPixelOn(x,y+4);
    setPixelOn(x,y+5);
    setPixelOn(x,y+6);
    setPixelOn(x,y+7);
    setPixelOn(x,y+8);
    setPixelOn(x,y+9);
    setPixelOn(x,y+10);
    setPixelOn(x,y+11);
    setPixelOn(x,y+12);
    setPixelOn(x,y+13);

    setPixelOn(x,y+13);
    setPixelOn(x+1,y+13);
    setPixelOn(x+2,y+13);
    setPixelOn(x+3,y+13);
    setPixelOn(x+4,y+13);
    setPixelOn(x+5,y+13);
    setPixelOn(x+6,y+13);
    setPixelOn(x+7,y+13);
}

// Writes an S with the upper left hand side corner at x,y
//width 8 px
void write_S(unsigned char x,unsigned char y){
    setPixelOn(x,y+2);
    setPixelOn(x,y+3);
    setPixelOn(x,y+4);
    setPixelOn(x,y+10);
    setPixelOn(x,y+11);
```

```
    setPixelOn(x+1,y+1);
    setPixelOn(x+1,y+5);
    setPixelOn(x+1,y+12);

    setPixelOn(x+2,y);
    setPixelOn(x+2,y+6);
    setPixelOn(x+2,y+13);

    setPixelOn(x+3,y);
    setPixelOn(x+3,y+6);
    setPixelOn(x+3,y+13);

    setPixelOn(x+4,y);
    setPixelOn(x+4,y+7);
    setPixelOn(x+4,y+13);

    setPixelOn(x+5,y);
    setPixelOn(x+5,y+7);
    setPixelOn(x+5,y+13);

    setPixelOn(x+6,y+1);
    setPixelOn(x+6,y+8);
    setPixelOn(x+6,y+12);

    setPixelOn(x+7,y+2);
    setPixelOn(x+7,y+3);
    setPixelOn(x+7,y+9);
    setPixelOn(x+7,y+10);
    setPixelOn(x+7,y+11);
}

// Writes an P with the upper left hand side corner at x,y
//width 8 px
void write_P(unsigned char x,unsigned char y){
    setPixelOn(x,y+1);
    setPixelOn(x,y+2);
    setPixelOn(x,y+3);
    setPixelOn(x,y+4);
    setPixelOn(x,y+5);
    setPixelOn(x,y+6);
    setPixelOn(x,y+7);
    setPixelOn(x,y+8);
    setPixelOn(x,y+9);
    setPixelOn(x,y+10);
    setPixelOn(x,y+11);
    setPixelOn(x,y+12);
    setPixelOn(x,y+13);

    setPixelOn(x+1,y);
    setPixelOn(x+1,y+6);

    setPixelOn(x+2,y);
    setPixelOn(x+2,y+6);

    setPixelOn(x+3,y);
    setPixelOn(x+3,y+6);

    setPixelOn(x+4,y);
    setPixelOn(x+4,y+6);
```

```
    setPixelOn(x+5,y);
    setPixelOn(x+5,y+6);

    setPixelOn(x+6,y+1);
    setPixelOn(x+6,y+5);

    setPixelOn(x+7,y+2);
    setPixelOn(x+7,y+3);
    setPixelOn(x+7,y+4);
}

// Writes an E with the upper left hand side corner at x,y
//width 7 px
void write_E(unsigned char x,unsigned char y){
    setPixelOn(x,y+1);
    setPixelOn(x,y+2);
    setPixelOn(x,y+3);
    setPixelOn(x,y+4);
    setPixelOn(x,y+5);
    setPixelOn(x,y+6);
    setPixelOn(x,y+7);
    setPixelOn(x,y+8);
    setPixelOn(x,y+9);
    setPixelOn(x,y+10);
    setPixelOn(x,y+11);
    setPixelOn(x,y+12);
    setPixelOn(x,y+13);

    setPixelOn(x,y+13);
    setPixelOn(x+1,y+13);
    setPixelOn(x+2,y+13);
    setPixelOn(x+3,y+13);
    setPixelOn(x+4,y+13);
    setPixelOn(x+5,y+13);
    setPixelOn(x+6,y+13);
    setPixelOn(x,y+13);
    setPixelOn(x+1,y+13);
    setPixelOn(x+2,y+13);
    setPixelOn(x+3,y+13);
    setPixelOn(x+4,y+13);
    setPixelOn(x+5,y+13);
    setPixelOn(x+6,y+13);

    setPixelOn(x,y+6);
    setPixelOn(x+1,y+6);
    setPixelOn(x+2,y+6);
    setPixelOn(x+3,y+6);
    setPixelOn(x+4,y+6);
    setPixelOn(x+5,y+6);
    setPixelOn(x+6,y+6);

    setPixelOn(x,y);
    setPixelOn(x+1,y);
    setPixelOn(x+2,y);
    setPixelOn(x+3,y);
    setPixelOn(x+4,y);
    setPixelOn(x+5,y);
    setPixelOn(x+6,y);
}

// Writes an R with the upper left hand side corner at x,y
```

```
//width 8 px
void write_R(unsigned char x,unsigned char y){
    setPixelOn(x,y+1);
    setPixelOn(x,y+2);
    setPixelOn(x,y+3);
    setPixelOn(x,y+4);
    setPixelOn(x,y+5);
    setPixelOn(x,y+6);
    setPixelOn(x,y+7);
    setPixelOn(x,y+8);
    setPixelOn(x,y+9);
    setPixelOn(x,y+10);
    setPixelOn(x,y+11);
    setPixelOn(x,y+12);
    setPixelOn(x,y+13);

    setPixelOn(x+1,y);
    setPixelOn(x+1,y+6);

    setPixelOn(x+2,y);
    setPixelOn(x+2,y+6);

    setPixelOn(x+3,y);
    setPixelOn(x+3,y+6);

    setPixelOn(x+4,y);
    setPixelOn(x+4,y+6);

    setPixelOn(x+5,y+1);
    setPixelOn(x+5,y+5);

    setPixelOn(x+6,y+2);
    setPixelOn(x+6,y+3);
    setPixelOn(x+6,y+4);

    setPixelOn(x+3,y+7);
    setPixelOn(x+4,y+8);
    setPixelOn(x+4,y+9);

    setPixelOn(x+5,y+10);
    setPixelOn(x+6,y+11);
    setPixelOn(x+6,y+12);
    setPixelOn(x+7,y+13);
}

// Writes an 1 with the upper left hand side corner at x,y
//width 4 px
void write_1(unsigned char x,unsigned char y){
    setPixelOn(x+1,y+2);
    setPixelOn(x+2,y+2);
    setPixelOn(x+3,y+1);

    setPixelOn(x+4,y);
    setPixelOn(x+4,y+1);
    setPixelOn(x+4,y+2);
    setPixelOn(x+4,y+3);
    setPixelOn(x+4,y+4);
    setPixelOn(x+4,y+5);
    setPixelOn(x+4,y+6);
    setPixelOn(x+4,y+7);
    setPixelOn(x+4,y+8);
```

```
    setPixelOn(x+4,y+9);
    setPixelOn(x+4,y+10);
    setPixelOn(x+4,y+11);
    setPixelOn(x+4,y+12);
    setPixelOn(x+4,y+13);
}

// Writes an 2 with the upper left hand side corner at x,y
//width 8 px
void write_2(unsigned char x,unsigned char y){
    setPixelOn(x,y+2);
    setPixelOn(x,y+3);
    setPixelOn(x,y+11);
    setPixelOn(x,y+12);
    setPixelOn(x,y+13);

    setPixelOn(x+1,y+1);
    setPixelOn(x+1,y+10);
    setPixelOn(x+1,y+13);

    setPixelOn(x+2,y);
    setPixelOn(x+2,y+9);
    setPixelOn(x+2,y+13);

    setPixelOn(x+2,y);
    setPixelOn(x+2,y+9);
    setPixelOn(x+2,y+13);

    setPixelOn(x+3,y);
    setPixelOn(x+3,y+8);
    setPixelOn(x+3,y+13);

    setPixelOn(x+4,y);
    setPixelOn(x+4,y+7);
    setPixelOn(x+4,y+13);

    setPixelOn(x+5,y);
    setPixelOn(x+5,y+6);
    setPixelOn(x+5,y+13);

    setPixelOn(x+6,y+1);
    setPixelOn(x+6,y+5);
    setPixelOn(x+6,y+13);

    setPixelOn(x+7,y+2);
    setPixelOn(x+7,y+3);
    setPixelOn(x+7,y+4);
    setPixelOn(x+7,y+13);
}

// Writes an M with the upper left hand side corner at x,y
void write_M(unsigned char x,unsigned char y){
    setPixelOn(x,y+13);
    setPixelOn(x,y+12);
    setPixelOn(x,y+11);
    setPixelOn(x,y+10);
    setPixelOn(x,y+9);
    setPixelOn(x,y+8);
    setPixelOn(x,y+7);
    setPixelOn(x,y+6);
    setPixelOn(x,y+5);
```

```
    setPixelOn(x,y+4);
    setPixelOn(x,y+3);
    setPixelOn(x,y+2);
    setPixelOn(x,y+1);
    setPixelOn(x,y);

    setPixelOn(x+1,y);
    setPixelOn(x+1,y+1);

    setPixelOn(x+2,y+2);
    setPixelOn(x+2,y+3);

    setPixelOn(x+3,y+4);
    setPixelOn(x+3,y+5);

    setPixelOn(x+4,y+6);
    setPixelOn(x+4,y+7);

    setPixelOn(x+5,y+8);
    setPixelOn(x+5,y+9);

    setPixelOn(x+6,y+10);
    setPixelOn(x+6,y+11);

    setPixelOn(x+7,y+12);
    setPixelOn(x+7,y+13);

    setPixelOn(x+13,y);
    setPixelOn(x+13,y+1);

    setPixelOn(x+12,y+2);
    setPixelOn(x+12,y+3);

    setPixelOn(x+11,y+4);
    setPixelOn(x+11,y+5);

    setPixelOn(x+10,y+6);
    setPixelOn(x+10,y+7);

    setPixelOn(x+9,y+8);
    setPixelOn(x+9,y+9);

    setPixelOn(x+8,y+10);
    setPixelOn(x+8,y+11);

    setPixelOn(x+14,y+13);
    setPixelOn(x+14,y+12);
    setPixelOn(x+14,y+11);
    setPixelOn(x+14,y+10);
    setPixelOn(x+14,y+9);
    setPixelOn(x+14,y+8);
    setPixelOn(x+14,y+7);
    setPixelOn(x+14,y+6);
    setPixelOn(x+14,y+5);
    setPixelOn(x+14,y+4);
    setPixelOn(x+14,y+3);
    setPixelOn(x+14,y+2);
    setPixelOn(x+14,y+1);
    setPixelOn(x+14,y);
}
```

```
void write_A_ring(unsigned char x,unsigned char y){
    setPixelOn(x+5,y-3);
    setPixelOn(x+5,y-4);

    setPixelOn(x+6,y-2);
    setPixelOn(x+6,y-5);

    setPixelOn(x+7,y-3);
    setPixelOn(x+7,y-4);
}

void write_exclamation(unsigned char x,unsigned char y){
    setPixelOn(x+1,y-1);
    setPixelOn(x+1,y);
    setPixelOn(x+1,y+1);
    setPixelOn(x+1,y+2);
    setPixelOn(x+1,y+3);
    setPixelOn(x+1,y+4);
    setPixelOn(x+1,y+5);
    setPixelOn(x+1,y+6);
    setPixelOn(x+1,y+7);
    setPixelOn(x+1,y+8);
    setPixelOn(x+1,y+9);
    setPixelOn(x+1,y+10);

    setPixelOn(x+1,y+12);
    setPixelOn(x+1,y+13);
    setPixelOn(x+1,y+14);

    setPixelOn(x,y+13);
    setPixelOn(x+2,y+13);
}

void unSelect(unsigned char x,unsigned char y)
{
 for(unsigned char k=0;k<11;k++){
    setPixelOff(x-1+k,y+17);
    setPixelOff(x-1+k,y+18);
 }
}

void write_sel(unsigned char x,unsigned char y)
{
 for(unsigned char k=0;k<11;k++){
    setPixelOn(x-1+k,y+17);
    setPixelOn(x-1+k,y+18);
 }
}

void splashScreen(){
    setPixelOn(10,39);
    setPixelOn(10,59);
    setPixelOn(11,39);
    setPixelOn(11,40);
    setPixelOn(11,41);
    setPixelOn(11,42);
    setPixelOn(11,43);
    setPixelOn(11,44);
    setPixelOn(11,45);
    setPixelOn(11,46);
    setPixelOn(11,47);
```

```
setPixelOn(11,58);
setPixelOn(11,59);
setPixelOn(12,43);
setPixelOn(12,44);
setPixelOn(12,45);
setPixelOn(12,46);
setPixelOn(12,47);
setPixelOn(12,48);
setPixelOn(12,49);
setPixelOn(12,50);
setPixelOn(12,51);
setPixelOn(12,52);
setPixelOn(12,53);
setPixelOn(12,54);
setPixelOn(12,55);
setPixelOn(12,56);
setPixelOn(12,57);
setPixelOn(12,58);
setPixelOn(12,59);
setPixelOn(13,37);
setPixelOn(13,38);
setPixelOn(13,39);
setPixelOn(13,40);
setPixelOn(13,41);
setPixelOn(13,42);
setPixelOn(13,43);
setPixelOn(13,44);
setPixelOn(13,45);
setPixelOn(13,46);
setPixelOn(13,47);
setPixelOn(13,48);
setPixelOn(13,49);
setPixelOn(13,50);
setPixelOn(13,51);
setPixelOn(13,52);
setPixelOn(13,53);
setPixelOn(13,54);
setPixelOn(13,55);
setPixelOn(13,56);
setPixelOn(13,57);
setPixelOn(13,58);
setPixelOn(14,37);
setPixelOn(14,38);
setPixelOn(14,39);
setPixelOn(14,40);
setPixelOn(14,41);
setPixelOn(14,42);
setPixelOn(14,43);
setPixelOn(15,38);
setPixelOn(15,39);
setPixelOn(15,40);
setPixelOn(16,39);
setPixelOn(16,40);
setPixelOn(16,41);
setPixelOn(16,42);
setPixelOn(17,40);
setPixelOn(17,41);
setPixelOn(17,42);
setPixelOn(17,43);
setPixelOn(17,44);
setPixelOn(18,43);
```

```
setPixelOn(18,44);
setPixelOn(18,45);
setPixelOn(18,46);
setPixelOn(19,42);
setPixelOn(19,43);
setPixelOn(19,44);
setPixelOn(20,38);
setPixelOn(20,39);
setPixelOn(20,40);
setPixelOn(20,41);
setPixelOn(20,42);
setPixelOn(20,43);
setPixelOn(21,38);
setPixelOn(21,39);
setPixelOn(21,40);
setPixelOn(21,41);
setPixelOn(22,39);
setPixelOn(22,40);
setPixelOn(22,41);
setPixelOn(22,42);
setPixelOn(22,43);
setPixelOn(22,44);
setPixelOn(22,45);
setPixelOn(22,46);
setPixelOn(23,43);
setPixelOn(23,44);
setPixelOn(23,45);
setPixelOn(23,46);
setPixelOn(23,47);
setPixelOn(23,48);
setPixelOn(23,49);
setPixelOn(23,50);
setPixelOn(23,51);
setPixelOn(23,52);
setPixelOn(23,53);
setPixelOn(23,54);
setPixelOn(23,55);
setPixelOn(23,56);
setPixelOn(23,57);
setPixelOn(23,58);
setPixelOn(24,7);
setPixelOn(24,8);
setPixelOn(24,9);
setPixelOn(24,10);
setPixelOn(24,11);
setPixelOn(24,12);
setPixelOn(24,13);
setPixelOn(24,14);
setPixelOn(24,15);
setPixelOn(24,16);
setPixelOn(24,17);
setPixelOn(24,18);
setPixelOn(24,19);
setPixelOn(24,20);
setPixelOn(24,21);
setPixelOn(24,22);
setPixelOn(24,23);
setPixelOn(24,24);
setPixelOn(24,25);
setPixelOn(24,26);
setPixelOn(24,27);
```

```
setPixelOn(24,49);
setPixelOn(24,50);
setPixelOn(24,51);
setPixelOn(24,52);
setPixelOn(24,53);
setPixelOn(24,54);
setPixelOn(24,55);
setPixelOn(24,56);
setPixelOn(25,7);
setPixelOn(25,27);
setPixelOn(26,7);
setPixelOn(26,27);
setPixelOn(27,7);
setPixelOn(27,27);
setPixelOn(28,7);
setPixelOn(28,11);
setPixelOn(28,12);
setPixelOn(28,13);
setPixelOn(28,14);
setPixelOn(28,18);
setPixelOn(28,19);
setPixelOn(28,20);
setPixelOn(28,21);
setPixelOn(28,22);
setPixelOn(28,23);
setPixelOn(28,24);
setPixelOn(28,25);
setPixelOn(28,26);
setPixelOn(28,27);
setPixelOn(28,46);
setPixelOn(28,47);
setPixelOn(28,48);
setPixelOn(28,49);
setPixelOn(28,50);
setPixelOn(28,51);
setPixelOn(29,7);
setPixelOn(29,11);
setPixelOn(29,14);
setPixelOn(29,18);
setPixelOn(29,40);
setPixelOn(29,41);
setPixelOn(29,42);
setPixelOn(29,43);
setPixelOn(29,44);
setPixelOn(29,45);
setPixelOn(29,46);
setPixelOn(29,47);
setPixelOn(29,48);
setPixelOn(29,49);
setPixelOn(29,50);
setPixelOn(29,51);
setPixelOn(29,52);
setPixelOn(30,7);
setPixelOn(30,11);
setPixelOn(30,14);
setPixelOn(30,18);
setPixelOn(30,40);
setPixelOn(30,41);
setPixelOn(30,42);
setPixelOn(30,43);
setPixelOn(30,44);
```

```
setPixelOn(30,45);
setPixelOn(30,51);
setPixelOn(30,52);
setPixelOn(31,7);
setPixelOn(31,11);
setPixelOn(31,14);
setPixelOn(31,18);
setPixelOn(31,40);
setPixelOn(31,41);
setPixelOn(31,51);
setPixelOn(31,52);
setPixelOn(32,8);
setPixelOn(32,12);
setPixelOn(32,13);
setPixelOn(32,17);
setPixelOn(32,40);
setPixelOn(32,41);
setPixelOn(32,50);
setPixelOn(32,51);
setPixelOn(33,8);
setPixelOn(33,17);
setPixelOn(33,39);
setPixelOn(33,40);
setPixelOn(33,41);
setPixelOn(33,49);
setPixelOn(33,50);
setPixelOn(34,9);
setPixelOn(34,16);
setPixelOn(34,39);
setPixelOn(34,40);
setPixelOn(34,41);
setPixelOn(34,48);
setPixelOn(34,49);
setPixelOn(35,10);
setPixelOn(35,15);
setPixelOn(35,39);
setPixelOn(35,40);
setPixelOn(35,47);
setPixelOn(35,48);
setPixelOn(35,49);
setPixelOn(36,11);
setPixelOn(36,12);
setPixelOn(36,13);
setPixelOn(36,14);
setPixelOn(36,39);
setPixelOn(36,40);
setPixelOn(36,46);
setPixelOn(36,47);
setPixelOn(37,39);
setPixelOn(37,40);
setPixelOn(37,46);
setPixelOn(37,47);
setPixelOn(38,39);
setPixelOn(38,40);
setPixelOn(38,45);
setPixelOn(38,46);
setPixelOn(39,14);
setPixelOn(39,15);
setPixelOn(39,16);
setPixelOn(39,17);
setPixelOn(39,18);
```

```
setPixelOn(39,19);
setPixelOn(39,20);
setPixelOn(39,39);
setPixelOn(39,40);
setPixelOn(39,41);
setPixelOn(39,42);
setPixelOn(39,43);
setPixelOn(39,44);
setPixelOn(39,45);
setPixelOn(40,12);
setPixelOn(40,13);
setPixelOn(40,21);
setPixelOn(40,22);
setPixelOn(40,39);
setPixelOn(40,40);
setPixelOn(40,41);
setPixelOn(40,42);
setPixelOn(40,43);
setPixelOn(40,44);
setPixelOn(40,45);
setPixelOn(40,46);
setPixelOn(40,47);
setPixelOn(41,11);
setPixelOn(41,23);
setPixelOn(41,45);
setPixelOn(41,46);
setPixelOn(41,47);
setPixelOn(42,10);
setPixelOn(42,24);
setPixelOn(42,47);
setPixelOn(42,48);
setPixelOn(42,49);
setPixelOn(43,9);
setPixelOn(43,25);
setPixelOn(43,48);
setPixelOn(43,49);
setPixelOn(43,50);
setPixelOn(44,8);
setPixelOn(44,26);
setPixelOn(44,49);
setPixelOn(44,50);
setPixelOn(44,51);
setPixelOn(45,8);
setPixelOn(45,26);
setPixelOn(45,38);
setPixelOn(45,39);
setPixelOn(45,40);
setPixelOn(45,41);
setPixelOn(45,42);
setPixelOn(46,7);
setPixelOn(46,27);
setPixelOn(46,40);
setPixelOn(46,41);
setPixelOn(46,42);
setPixelOn(46,43);
setPixelOn(47,7);
setPixelOn(47,27);
setPixelOn(47,42);
setPixelOn(47,43);
setPixelOn(47,44);
setPixelOn(47,45);
```

```
setPixelOn(47,46);
setPixelOn(48,7);
setPixelOn(48,27);
setPixelOn(48,28);
setPixelOn(48,29);
setPixelOn(48,30);
setPixelOn(48,31);
setPixelOn(48,32);
setPixelOn(48,33);
setPixelOn(48,34);
setPixelOn(48,35);
setPixelOn(48,36);
setPixelOn(48,44);
setPixelOn(48,45);
setPixelOn(48,46);
setPixelOn(48,47);
setPixelOn(48,48);
setPixelOn(48,49);
setPixelOn(48,57);
setPixelOn(48,58);
setPixelOn(48,59);
setPixelOn(49,7);
setPixelOn(49,27);
setPixelOn(49,28);
setPixelOn(49,29);
setPixelOn(49,30);
setPixelOn(49,31);
setPixelOn(49,32);
setPixelOn(49,33);
setPixelOn(49,34);
setPixelOn(49,35);
setPixelOn(49,36);
setPixelOn(49,46);
setPixelOn(49,47);
setPixelOn(49,48);
setPixelOn(49,49);
setPixelOn(49,50);
setPixelOn(49,51);
setPixelOn(49,52);
setPixelOn(49,53);
setPixelOn(49,54);
setPixelOn(49,55);
setPixelOn(49,56);
setPixelOn(49,57);
setPixelOn(49,58);
setPixelOn(50,7);
setPixelOn(50,27);
setPixelOn(50,28);
setPixelOn(50,29);
setPixelOn(50,30);
setPixelOn(50,31);
setPixelOn(50,32);
setPixelOn(50,33);
setPixelOn(50,34);
setPixelOn(50,35);
setPixelOn(50,36);
setPixelOn(50,49);
setPixelOn(50,50);
setPixelOn(50,51);
setPixelOn(50,52);
setPixelOn(50,53);
```

```
setPixelOn(50,54);
setPixelOn(50,55);
setPixelOn(50,56);
setPixelOn(50,57);
setPixelOn(51,7);
setPixelOn(51,27);
setPixelOn(51,28);
setPixelOn(51,29);
setPixelOn(51,30);
setPixelOn(51,31);
setPixelOn(51,32);
setPixelOn(51,33);
setPixelOn(51,34);
setPixelOn(51,35);
setPixelOn(51,36);
setPixelOn(51,50);
setPixelOn(51,51);
setPixelOn(51,52);
setPixelOn(52,7);
setPixelOn(52,27);
setPixelOn(52,49);
setPixelOn(52,50);
setPixelOn(52,51);
setPixelOn(53,8);
setPixelOn(53,26);
setPixelOn(53,46);
setPixelOn(53,47);
setPixelOn(53,48);
setPixelOn(53,49);
setPixelOn(53,50);
setPixelOn(54,8);
setPixelOn(54,26);
setPixelOn(54,44);
setPixelOn(54,45);
setPixelOn(54,46);
setPixelOn(54,47);
setPixelOn(54,48);
setPixelOn(55,9);
setPixelOn(55,25);
setPixelOn(55,43);
setPixelOn(55,44);
setPixelOn(55,45);
setPixelOn(56,10);
setPixelOn(56,24);
setPixelOn(56,41);
setPixelOn(56,42);
setPixelOn(56,43);
setPixelOn(57,11);
setPixelOn(57,23);
setPixelOn(57,40);
setPixelOn(57,41);
setPixelOn(58,12);
setPixelOn(58,13);
setPixelOn(58,21);
setPixelOn(58,22);
setPixelOn(58,38);
setPixelOn(58,39);
setPixelOn(58,40);
setPixelOn(59,14);
setPixelOn(59,15);
setPixelOn(59,16);
```

```
setPixelOn(59,17);
setPixelOn(59,18);
setPixelOn(59,19);
setPixelOn(59,20);
setPixelOn(61,32);
setPixelOn(61,57);
setPixelOn(61,58);
setPixelOn(62,7);
setPixelOn(62,8);
setPixelOn(62,9);
setPixelOn(62,10);
setPixelOn(62,11);
setPixelOn(62,12);
setPixelOn(62,13);
setPixelOn(62,14);
setPixelOn(62,15);
setPixelOn(62,16);
setPixelOn(62,17);
setPixelOn(62,18);
setPixelOn(62,19);
setPixelOn(62,20);
setPixelOn(62,21);
setPixelOn(62,22);
setPixelOn(62,23);
setPixelOn(62,24);
setPixelOn(62,25);
setPixelOn(62,26);
setPixelOn(62,27);
setPixelOn(62,32);
setPixelOn(62,33);
setPixelOn(62,34);
setPixelOn(62,35);
setPixelOn(62,36);
setPixelOn(62,37);
setPixelOn(62,56);
setPixelOn(62,57);
setPixelOn(63,7);
setPixelOn(63,27);
setPixelOn(63,34);
setPixelOn(63,35);
setPixelOn(63,36);
setPixelOn(63,37);
setPixelOn(63,38);
setPixelOn(63,39);
setPixelOn(63,40);
setPixelOn(63,41);
setPixelOn(63,42);
setPixelOn(63,43);
setPixelOn(63,44);
setPixelOn(63,45);
setPixelOn(63,46);
setPixelOn(63,47);
setPixelOn(63,48);
setPixelOn(63,49);
setPixelOn(63,50);
setPixelOn(63,51);
setPixelOn(63,52);
setPixelOn(63,53);
setPixelOn(63,54);
setPixelOn(63,55);
setPixelOn(63,56);
```

```
setPixelOn(64,7);
setPixelOn(64,27);
setPixelOn(64,40);
setPixelOn(64,41);
setPixelOn(64,42);
setPixelOn(64,43);
setPixelOn(64,44);
setPixelOn(64,45);
setPixelOn(64,46);
setPixelOn(64,47);
setPixelOn(64,48);
setPixelOn(64,49);
setPixelOn(64,50);
setPixelOn(64,51);
setPixelOn(64,52);
setPixelOn(64,53);
setPixelOn(64,54);
setPixelOn(65,7);
setPixelOn(65,15);
setPixelOn(65,27);
setPixelOn(65,42);
setPixelOn(65,43);
setPixelOn(65,44);
setPixelOn(65,45);
setPixelOn(66,7);
setPixelOn(66,14);
setPixelOn(66,16);
setPixelOn(66,17);
setPixelOn(66,18);
setPixelOn(66,19);
setPixelOn(66,20);
setPixelOn(66,21);
setPixelOn(66,22);
setPixelOn(66,23);
setPixelOn(66,24);
setPixelOn(66,25);
setPixelOn(66,26);
setPixelOn(66,27);
setPixelOn(66,42);
setPixelOn(66,43);
setPixelOn(66,44);
setPixelOn(67,7);
setPixelOn(67,15);
setPixelOn(67,42);
setPixelOn(67,43);
setPixelOn(68,7);
setPixelOn(68,16);
setPixelOn(68,17);
setPixelOn(68,41);
setPixelOn(68,42);
setPixelOn(68,43);
setPixelOn(69,7);
setPixelOn(69,8);
setPixelOn(69,18);
setPixelOn(69,19);
setPixelOn(69,41);
setPixelOn(69,42);
setPixelOn(70,9);
setPixelOn(70,10);
setPixelOn(70,20);
setPixelOn(70,21);
```

```
setPixelOn(70,41);
setPixelOn(70,42);
setPixelOn(71,11);
setPixelOn(71,12);
setPixelOn(71,22);
setPixelOn(71,23);
setPixelOn(71,41);
setPixelOn(71,42);
setPixelOn(72,13);
setPixelOn(72,14);
setPixelOn(72,24);
setPixelOn(72,25);
setPixelOn(72,41);
setPixelOn(72,42);
setPixelOn(73,15);
setPixelOn(73,16);
setPixelOn(73,26);
setPixelOn(73,27);
setPixelOn(73,41);
setPixelOn(73,42);
setPixelOn(73,43);
setPixelOn(74,17);
setPixelOn(74,27);
setPixelOn(74,41);
setPixelOn(74,42);
setPixelOn(74,43);
setPixelOn(75,7);
setPixelOn(75,8);
setPixelOn(75,9);
setPixelOn(75,10);
setPixelOn(75,11);
setPixelOn(75,12);
setPixelOn(75,13);
setPixelOn(75,14);
setPixelOn(75,15);
setPixelOn(75,16);
setPixelOn(75,18);
setPixelOn(75,27);
setPixelOn(75,42);
setPixelOn(75,43);
setPixelOn(75,44);
setPixelOn(75,45);
setPixelOn(75,46);
setPixelOn(75,47);
setPixelOn(75,48);
setPixelOn(76,7);
setPixelOn(76,17);
setPixelOn(76,27);
setPixelOn(76,44);
setPixelOn(76,45);
setPixelOn(76,46);
setPixelOn(76,47);
setPixelOn(76,48);
setPixelOn(76,49);
setPixelOn(76,50);
setPixelOn(76,51);
setPixelOn(76,52);
setPixelOn(76,53);
setPixelOn(76,54);
setPixelOn(76,55);
setPixelOn(77,7);
```

```
setPixelOn(77,27);
setPixelOn(77,54);
setPixelOn(77,55);
setPixelOn(77,56);
setPixelOn(77,57);
setPixelOn(77,58);
setPixelOn(78,7);
setPixelOn(78,27);
setPixelOn(78,57);
setPixelOn(78,58);
setPixelOn(79,7);
setPixelOn(79,8);
setPixelOn(79,9);
setPixelOn(79,10);
setPixelOn(79,11);
setPixelOn(79,12);
setPixelOn(79,13);
setPixelOn(79,14);
setPixelOn(79,15);
setPixelOn(79,16);
setPixelOn(79,17);
setPixelOn(79,18);
setPixelOn(79,19);
setPixelOn(79,20);
setPixelOn(79,21);
setPixelOn(79,22);
setPixelOn(79,23);
setPixelOn(79,24);
setPixelOn(79,25);
setPixelOn(79,26);
setPixelOn(79,27);
setPixelOn(79,58);
setPixelOn(79,59);
setPixelOn(81,41);
setPixelOn(81,42);
setPixelOn(81,43);
setPixelOn(81,44);
setPixelOn(81,45);
setPixelOn(81,46);
setPixelOn(81,47);
setPixelOn(81,48);
setPixelOn(81,49);
setPixelOn(82,14);
setPixelOn(82,15);
setPixelOn(82,16);
setPixelOn(82,17);
setPixelOn(82,18);
setPixelOn(82,19);
setPixelOn(82,20);
setPixelOn(82,40);
setPixelOn(82,41);
setPixelOn(82,42);
setPixelOn(82,43);
setPixelOn(82,47);
setPixelOn(82,48);
setPixelOn(82,49);
setPixelOn(82,50);
setPixelOn(83,12);
setPixelOn(83,13);
setPixelOn(83,21);
setPixelOn(83,22);
```

```
setPixelOn(83,39);
setPixelOn(83,40);
setPixelOn(83,41);
setPixelOn(83,46);
setPixelOn(83,49);
setPixelOn(83,50);
setPixelOn(84,11);
setPixelOn(84,23);
setPixelOn(84,39);
setPixelOn(84,40);
setPixelOn(84,41);
setPixelOn(84,45);
setPixelOn(84,46);
setPixelOn(84,50);
setPixelOn(84,51);
setPixelOn(84,52);
setPixelOn(85,10);
setPixelOn(85,24);
setPixelOn(85,39);
setPixelOn(85,40);
setPixelOn(85,45);
setPixelOn(85,50);
setPixelOn(85,51);
setPixelOn(85,52);
setPixelOn(86,9);
setPixelOn(86,25);
setPixelOn(86,38);
setPixelOn(86,39);
setPixelOn(86,44);
setPixelOn(86,51);
setPixelOn(86,52);
setPixelOn(87,8);
setPixelOn(87,14);
setPixelOn(87,15);
setPixelOn(87,16);
setPixelOn(87,17);
setPixelOn(87,18);
setPixelOn(87,19);
setPixelOn(87,20);
setPixelOn(87,26);
setPixelOn(87,38);
setPixelOn(87,39);
setPixelOn(87,43);
setPixelOn(87,44);
setPixelOn(87,51);
setPixelOn(87,52);
setPixelOn(88,8);
setPixelOn(88,13);
setPixelOn(88,21);
setPixelOn(88,26);
setPixelOn(88,38);
setPixelOn(88,39);
setPixelOn(88,42);
setPixelOn(88,43);
setPixelOn(88,44);
setPixelOn(88,52);
setPixelOn(89,7);
setPixelOn(89,12);
setPixelOn(89,22);
setPixelOn(89,27);
setPixelOn(89,38);
```

```
setPixelOn(89,39);
setPixelOn(89,41);
setPixelOn(89,42);
setPixelOn(89,43);
setPixelOn(89,52);
setPixelOn(90,7);
setPixelOn(90,11);
setPixelOn(90,23);
setPixelOn(90,27);
setPixelOn(90,38);
setPixelOn(90,39);
setPixelOn(90,40);
setPixelOn(90,41);
setPixelOn(90,42);
setPixelOn(90,51);
setPixelOn(90,52);
setPixelOn(91,7);
setPixelOn(91,11);
setPixelOn(91,23);
setPixelOn(91,27);
setPixelOn(91,39);
setPixelOn(91,40);
setPixelOn(91,51);
setPixelOn(91,52);
setPixelOn(92,7);
setPixelOn(92,11);
setPixelOn(92,23);
setPixelOn(92,27);
setPixelOn(92,51);
setPixelOn(93,7);
setPixelOn(93,11);
setPixelOn(93,23);
setPixelOn(93,27);
setPixelOn(93,51);
setPixelOn(94,7);
setPixelOn(94,11);
setPixelOn(94,19);
setPixelOn(94,20);
setPixelOn(94,21);
setPixelOn(94,23);
setPixelOn(94,27);
setPixelOn(94,50);
setPixelOn(94,51);
setPixelOn(95,7);
setPixelOn(95,11);
setPixelOn(95,19);
setPixelOn(95,21);
setPixelOn(95,23);
setPixelOn(95,27);
setPixelOn(95,49);
setPixelOn(95,50);
setPixelOn(96,8);
setPixelOn(96,12);
setPixelOn(96,19);
setPixelOn(96,21);
setPixelOn(96,22);
setPixelOn(96,26);
setPixelOn(96,48);
setPixelOn(96,49);
setPixelOn(96,50);
setPixelOn(97,8);
```

```
setPixelOn(97,13);
setPixelOn(97,19);
setPixelOn(97,26);
setPixelOn(97,47);
setPixelOn(97,48);
setPixelOn(97,49);
setPixelOn(98,9);
setPixelOn(98,14);
setPixelOn(98,15);
setPixelOn(98,19);
setPixelOn(98,25);
setPixelOn(98,47);
setPixelOn(98,48);
setPixelOn(98,49);
setPixelOn(99,10);
setPixelOn(99,15);
setPixelOn(99,19);
setPixelOn(99,24);
setPixelOn(100,11);
setPixelOn(100,15);
setPixelOn(100,19);
setPixelOn(100,23);
setPixelOn(100,39);
setPixelOn(100,57);
setPixelOn(101,12);
setPixelOn(101,13);
setPixelOn(101,15);
setPixelOn(101,19);
setPixelOn(101,21);
setPixelOn(101,22);
setPixelOn(101,39);
setPixelOn(101,40);
setPixelOn(101,41);
setPixelOn(101,42);
setPixelOn(101,43);
setPixelOn(101,44);
setPixelOn(101,45);
setPixelOn(101,46);
setPixelOn(101,47);
setPixelOn(101,56);
setPixelOn(101,57);
setPixelOn(102,14);
setPixelOn(102,15);
setPixelOn(102,19);
setPixelOn(102,20);
setPixelOn(102,43);
setPixelOn(102,44);
setPixelOn(102,45);
setPixelOn(102,46);
setPixelOn(102,47);
setPixelOn(102,48);
setPixelOn(102,49);
setPixelOn(102,50);
setPixelOn(102,51);
setPixelOn(102,52);
setPixelOn(102,53);
setPixelOn(102,54);
setPixelOn(102,55);
setPixelOn(103,37);
setPixelOn(103,38);
setPixelOn(103,39);
```

```
setPixelOn(103,40);
setPixelOn(103,41);
setPixelOn(103,42);
setPixelOn(103,43);
setPixelOn(103,44);
setPixelOn(103,45);
setPixelOn(103,46);
setPixelOn(103,47);
setPixelOn(103,48);
setPixelOn(103,49);
setPixelOn(103,50);
setPixelOn(103,51);
setPixelOn(103,52);
setPixelOn(103,53);
setPixelOn(103,54);
setPixelOn(104,37);
setPixelOn(104,38);
setPixelOn(104,39);
setPixelOn(104,40);
setPixelOn(105,38);
setPixelOn(105,39);
setPixelOn(106,39);
setPixelOn(106,40);
setPixelOn(106,41);
setPixelOn(107,40);
setPixelOn(107,41);
setPixelOn(107,42);
setPixelOn(107,43);
setPixelOn(107,44);
setPixelOn(108,42);
setPixelOn(108,43);
setPixelOn(108,44);
setPixelOn(108,45);
setPixelOn(108,46);
setPixelOn(109,42);
setPixelOn(109,43);
setPixelOn(109,44);
setPixelOn(109,45);
setPixelOn(109,46);
setPixelOn(109,47);
setPixelOn(110,38);
setPixelOn(110,39);
setPixelOn(110,40);
setPixelOn(110,41);
setPixelOn(110,42);
setPixelOn(110,43);
setPixelOn(111,38);
setPixelOn(111,39);
setPixelOn(111,40);
setPixelOn(111,41);
setPixelOn(111,42);
setPixelOn(112,39);
setPixelOn(112,40);
setPixelOn(112,41);
setPixelOn(112,42);
setPixelOn(112,43);
setPixelOn(112,44);
setPixelOn(112,45);
setPixelOn(112,46);
setPixelOn(112,47);
setPixelOn(112,48);
```

```
    setPixelOn(112,49);
    setPixelOn(112,57);
    setPixelOn(113,43);
    setPixelOn(113,44);
    setPixelOn(113,45);
    setPixelOn(113,46);
    setPixelOn(113,47);
    setPixelOn(113,48);
    setPixelOn(113,49);
    setPixelOn(113,50);
    setPixelOn(113,51);
    setPixelOn(113,52);
    setPixelOn(113,53);
    setPixelOn(113,54);
    setPixelOn(113,55);
    setPixelOn(113,56);
    setPixelOn(113,57);
    setPixelOn(113,58);
    setPixelOn(114,49);
    setPixelOn(114,50);
    setPixelOn(114,51);
    setPixelOn(114,52);
    setPixelOn(114,53);
    setPixelOn(114,54);
    setPixelOn(114,55);
    setPixelOn(114,56);
}
```