```
                        #include <msp430.h>
;-------------------------------------------------------------CONSTANTS-------------------------------------------------------
RED_LED                 EQU     BIT7
GREEN_LED               EQU     BIT6
YELLOW_LED              EQU     BIT5
RED_LED2                EQU     BIT4
RED_LED_B               EQU     BIT3
GREEN_LED_B             EQU     BIT2
YELLOW_LED_B            EQU     BIT1
RED_LED2_B              EQU     BIT0
;-------------------------------------------------------------MACROS----------------------------------------------------------
TurnOnLED       MACRO           LED
                bis.b   LED, &P1OUT                                     ;Turns on corresponding LED
                ENDM
;-------------------------------------------------------------------------------------------------------------------------------
TurnOffLED      MACRO           LED
                bic.b   LED, &P1OUT                                     ;Turns off corresponding LED
                mov     #0, TA0CCR0                                     ;Turns off buzzer
                ENDM
;-------------------------------------------------------------------------------------------------------------------------------
Delay           MACRO           delayTime
                local   Repeat
                mov     delayTime, R4                                   ;Sets up delay
Repeat:         dec     R4                                              ;Decreases the delay counter
                jnz     Repeat                                          ;Delay Over?
                ENDM
;-------------------------------------------------------------------------------------------------------------------------------
Sound           MACRO           value
                mov     value, TA0CCR0                                  ;Changes the frequency of the PWM output
                ENDM
;-------------------------------------------------------------------------------------------------------------------------------
Multiply        MACRO           value1, value2                          ;Peasant's Algorithm
                        local   A_Even_Init, Start_Mult, Finish
                        mov     value1, R4
                        mov     value2, R5


                        bit.b   #BIT0, R4
                        jnc     A_Even_Init
                        mov     R5,R6
```

```
                        jmp     Start_Mult
A_Even_Init:            mov     #0, R6
Start_Mult:             cmp     #1, R4
                        jlo     Finish
                        clrc
                        rrc     R4
                        rla     R5
                        bit.b   #BIT0, R4
                        jnc     Start_Mult
                        add     R5,R6
                        jmp     Start_Mult
Finish:
                        ENDM
;-------------------------------------------------------------------------------------------------------------------------------
RNG             MACRO                                                   ;Linear Congruential Generator Algorithm
                Multiply R15, 48271
                mov             R6, R15
                ENDM

;-------------------------------------------------------------------------------------------------------------------------------
GenerateRound   MACRO           numberOfLEDs, delay
                local   ShowPattern, Loop,RED_LED2_ON, GREEN_LED_ON, RED_LED_ON, YELLOW_LED_ON, CheckLoop, SoundRed,
SoundGreen, SoundYellow, SoundRed2
                RNG                                                     ;Generate random number and stores on R15
                mov     numberOfLEDs, R5                                ;Loop count (number of LEDs)
                mov     #0, R6                                          ;R6=0, used for generating the number
Loop:           rra     R15                                             ;Shifts the LSB of number...
                rlc     R6                                              ;...into R6
                rra     R15                                             ;Shifts another LSB...
                rlc     R6                                              ;...into R6
                dec     R5                                              ;Decrease R5
                jnz     Loop                                            ;Loop over?
                mov     numberOfLEDs, R5                                ;Loop counter for showing the pattern
                mov     #0, R8
ShowPattern:    mov     #0, R7                                          ;R7=0, used for saving the pattern generated
                rla     R8                                              ;Prepares R8 for next set of inputs
                rla     R8
                rla     R8
                rla     R8
                rra     R6                                              ;Shifts LSB...
```

```
                rlc     R7                      ;...into R7
                rra     R6                      ;Shifts LSB again...
                rlc     R7                      ;...into R7
                cmp     #0, R7                  ;Compares with 0 to determine which LED to turn on
                jeq     RED_LED2_ON
                cmp     #1, R7                  ;Compares with 1 to determine which LED to turn on
                jeq     YELLOW_LED_ON
                cmp     #2, R7                  ;Compares with 2 to determine which LED to turn on
                jeq     GREEN_LED_ON

RED_LED_ON:     cmp             #1, R13
                jeq             SoundRed
                TurnOnLED       #RED_LED
                Sound           #1900
                Delay           delay
                TurnOffLED      #RED_LED
                Delay           delay
                add             #8h, R8
                jmp             CheckLoop

SoundRed:       Sound           #1900
                Delay           delay
                Sound           #0
                Delay           delay
                add             #8h, R8
                jmp             CheckLoop

RED_LED2_ON:    cmp             #1, R13
                jeq             SoundRed2
                TurnOnLED       #RED_LED2
                Sound           #950
                Delay           delay
                TurnOffLED      #RED_LED2
                Delay           delay
                add             #1h, R8
                jmp             CheckLoop

SoundRed2:      Sound           #950
                Delay           delay
                Sound           #0
```

```
                Delay       delay
                add         #1h, R8
                jmp         CheckLoop

GREEN_LED_ON:   cmp         #1, R13
                jeq         SoundGreen
                TurnOnLED   #GREEN_LED
                Sound       #1500
                Delay       delay
                TurnOffLED  #GREEN_LED
                Delay       delay
                add         #4h, R8
                jmp         CheckLoop

SoundGreen:     Sound       #1500
                Delay       delay
                Sound       #0
                Delay       delay
                add         #4h, R8
                jmp         CheckLoop

YELLOW_LED_ON:  cmp         #1, R13
                jeq         SoundYellow
                TurnOnLED   #YELLOW_LED
                Sound       #1275
                Delay       delay
                TurnOffLED  #YELLOW_LED
                Delay       delay
                add         #2h, R8
                jmp         CheckLoop

SoundYellow:    Sound       #1275
                Delay       delay
                Sound       #0
                Delay       delay
                add         #2h, R8
                jmp         CheckLoop

CheckLoop:      dec         R5          ;Decreases R5 (loop count)
                jnz         ShowPattern ;Loop over?
```

```
                ENDM
;-------------------------------------------------------------------------------------------------------------------------------
CORRECT             MACRO
                TurnOnLED       #RED_LED                                ;Turns on and off LEDs in a shifting pattern
                Sound           #1900
                Delay           #30000
                TurnOffLED      #RED_LED
                TurnOnLED       #GREEN_LED
                Sound           #1500
                Delay           #30000
                TurnOffLED      #GREEN_LED
                TurnOnLED       #YELLOW_LED
                Sound           #1275
                Delay           #30000
                TurnOffLED      #YELLOW_LED
                TurnOnLED       #RED_LED2
                Sound           #950
                Delay           #30000
                TurnOffLED      #RED_LED2
                TurnOnLED       #YELLOW_LED
                Sound           #1275
                Delay           #30000
                TurnOffLED      #YELLOW_LED
                TurnOnLED       #GREEN_LED
                Sound           #1500
                Delay           #30000
                TurnOffLED      #GREEN_LED
                TurnOnLED       #RED_LED
                Sound           #1900
                Delay           #30000
                TurnOffLED      #RED_LED
                Delay           #65535
                Delay           #65535
                ENDM
;-------------------------------------------------------------------------------------------------------------------------------
READ_INPUT  MACRO           numberOfLEDs
                local   POLL, INDIV_TEST, TurnOnRedLED, TurnOnGreenLED, TurnOnYellowLED, POLL_TURNOFF, LED_OFF, CMP_INPUT,
CorrectPattern
                mov     #0, R10
                mov     #0, R12
```

```
                mov     numberOfLEDs, R5                                        ;Loop counter
POLL:           bit.b   #RED_LED_B+GREEN_LED_B+YELLOW_LED_B+RED_LED2_B, &P2IN ;Test all inputs
                jc      INDIV_TEST
                bit.b   #BIT3, &P1IN
                jnc     START                                                   ;Resets the game if the P1.3 button is pressed
                jmp     POLL
INDIV_TEST:     add.b   P2IN, R10                                               ;Saves input to R10
                swpb    R10                                                     ;Swaps bytes to get the LSB (the button pressed)
                rla     R10                                                     ;Discard the MSNibble of the byte
                rla     R10
                rla     R10
                rla     R10
                rla     R10                                                     ;Capture the MSB on the CF...
                rlc     R12                                                     ;...into R12
                rla     R10
                rlc     R12
                rla     R10
                rlc     R12
                rla     R10
                rlc     R12                                                     ;R12 = Button pressed


                bit.b   #RED_LED_B, &P2IN                                       ;Tests which button was pressed and turns on the corresponding LED
                jc      TurnOnRedLED
                bit.b   #GREEN_LED_B, &P2IN
                jc      TurnOnGreenLED
                bit.b   #YELLOW_LED_B, &P2IN
                jc      TurnOnYellowLED
                TurnOnLED       #RED_LED2
                Sound   #950
                jmp POLL_TURNOFF
TurnOnRedLED:   TurnOnLED       #RED_LED
                Sound   #1900
                jmp POLL_TURNOFF
TurnOnGreenLED:         TurnOnLED       #GREEN_LED
                Sound   #1500
                jmp POLL_TURNOFF
TurnOnYellowLED:TurnOnLED   #YELLOW_LED
                Sound   #1275
                jmp POLL_TURNOFF
```

```
POLL_TURNOFF:       bit.b   #RED_LED_B+GREEN_LED_B+YELLOW_LED_B+RED_LED2_B, &P2IN ;Test the input for low (button released) and
turns off the LED
                    jc      POLL_TURNOFF
LED_OFF:            TurnOffLED      #RED_LED
                    TurnOffLED      #GREEN_LED
                    TurnOffLED      #YELLOW_LED
                    TurnOffLED      #RED_LED2
                    Delay   #50000
                    dec     R5
                    jnz     POLL

CMP_INPUT:          cmp     R8, R12                                 ;Compares the input generated by the MCU to the input entered by the
player
                    jeq     CorrectPattern                          ;If they're equal, notify that it was correct
                    TurnOnLED       #RED_LED                        ;Turns on one red LED...
                    TurnOnLED       #RED_LED2                       ;...and the other red LED...
                    Sound           #10000                          ;...to notify that the sequence entered was wrong
                    Delay           #65535                          ;Wait
                    Delay           #65535                          ;Wait
                    Delay           #65535                          ;Wait
                    TurnOffLED      #RED_LED                        ;Turn off one red LED...
                    TurnOffLED      #RED_LED2                       ;...and the other red LED
                    Delay           #65535                          ;Wait
                    jmp             START                           ;Restart the game if the input was wrong
CorrectPattern:     Correct                                         ;Notify that the input was correct
                    ENDM
;-------------------------------------------------------------------------------------------------------------------------
START:  ORG     0C000h
        mov     #WDTPW+WDTHOLD, &WDTCTL                                         ;Stop WDT
        mov.b   #0, &P1OUT                                      ;All LEDs off
        mov.b   #0xF7, &P1DIR                           ;Set P1.7-4 as outputs, P1.2 as output, P1.3 as input, unused pins as output
        mov.b   #0xF0, &P2DIR                                           ;Set P2.3-0 as inputs, unused pins as output
        bis.b   #RED_LED_B+GREEN_LED_B+YELLOW_LED_B+RED_LED2_B, &P2REN          ;Enable Internal Resistors
        bic.b   #RED_LED_B+GREEN_LED_B+YELLOW_LED_B+RED_LED2_B, &P2OUT          ;Internal Resistors act as Pull-Down Resistors
        bis.b   #BIT3, &P1REN                                   ;Enable internal resistor on P1.3...
        bis.b   #BIT3, &P1OUT                                   ;...and set it as a pull-up resistor.
;-----------------------------------------------------------TIMER SETUP FOR PWM-------------------------------------------------
PWM_SETUP:  bis.b   #BIT2, &P1SEL                               ;Set P1.2 as PWM Output
            mov     #0, TA0CCR0                                 ;"Frequencies": C=1900 E=1500 G = 1275 C = 950
```

```
                mov     #OUTMOD_7, TA0CCTL1                                     ;Output mode: Set/Reset
                mov     #500, TA0CCR1                                           ;Set the duty cycle to 50%
                mov     #TASSEL_2+MC_1, TA0CTL          ;SMCLK as input, counter in count-up mode (counts up to the value of TA0CCR0)
;-------------------------------------------------------------START THE GAME-------------------------------------------------------------
StartGame:      TurnOnLED       #RED_LED                                        ;Notify that the game has been turned on
                Sound   #1900
                Delay   #65535
                TurnOffLED      #RED_LED
                TurnOnLED       #GREEN_LED
                Delay   #65535
                TurnOffLED      #GREEN_LED
                TurnOnLED       #YELLOW_LED
                Sound   #950
                Delay   #65535
                TurnOffLED      #YELLOW_LED
                TurnOnLED       #RED_LED2
                Delay   #65535
                TurnOffLED      #RED_LED2


SelectDiff:     bit.b   #BIT3, &P1IN                                            ;Check if P1.3 is pressed, in order to reset.
                jnc     StartGame
                bit.b   #RED_LED_B+GREEN_LED_B+YELLOW_LED_B, &P2IN      ;Check if any button to select difficulty has been pressed
                jnc     SelectDiff
                bit.b   #RED_LED_B, &P2IN
                jc      NormalMode
                bit.b   #GREEN_LED_B, &P2IN
                jc      HardMode
ExpertMode:     mov     #1, R13                                                 ;If R13 = 0 => LEDs turn on , if R13 = 1, sounds only
                mov     #60000, R14                                             ;Delay between LEDs on R14
                jmp     Game
NormalMode:     mov     #0, R13
                mov     #60000, R14
                jmp     Game
HardMode:       mov     #0, R13
                mov     #40000, R14

Game:           TurnOnLED       #0F0h
                Sound           #950
                Delay           #65535
```

```
                TurnOffLED      #0F0h
                Delay           #65535
                Delay           #65535
                Delay           #65535
                mov             #1, R9          ;Number of LEDs for current round
RND_CHANGE:     mov             #5, R11         ;Number of rounds with R9's amount of LEDs
RepeatRound:    GenerateRound   R9, R14         ;Sets up the round and shows the pattern
                READ_INPUT      R9              ;Reads the input
                dec             R11             ;Counter decreases
                jnz             RepeatRound     ;Jump if 5 rounds with the same amount of LEDs haven't passed
                add             #1, R9          ;Increment the amound of LEDs in a round by 1
                cmp             #5, R9          ;Check if the amount of LEDs is at most 4
                jeq             Finish
                jmp             RND_CHANGE      ;If not, change the round with the incremented amount of LEDs


Finish:         TurnOnLED       #RED_LED        ;Notifies that the player won all rounds!
                TurnOnLED       #GREEN_LED
                TurnOnLED       #YELLOW_LED
                TurnOnLED       #RED_LED2
                Sound           #950
                Delay           #50000
                TurnOffLED      #RED_LED
                TurnOffLED      #GREEN_LED
                TurnOffLED      #YELLOW_LED
                TurnOffLED      #RED_LED2
                Delay           #10000
                TurnOnLED       #RED_LED
                TurnOnLED       #GREEN_LED
                TurnOnLED       #YELLOW_LED
                TurnOnLED       #RED_LED2
                Sound           #950
                Delay           #65535
                Delay           #65535
                Delay           #65535
                TurnOffLED      #RED_LED
                TurnOffLED      #GREEN_LED
                TurnOffLED      #YELLOW_LED
                TurnOffLED      #RED_LED2
                Delay           #65535
```

```
                Delay           #65535
                Delay           #65535
;-------------------------------------------------------------------------------------------------------------------------------
                ORG 0FFFEh
                DW  START
                END
```

Register Map:
R4 - Used in the Delay macro as the counter, used during multiplication macro
R5 - Used as loop counter when needed, used during multiplication macro
R6 - Intermediary for generating the random sequence (gets groups of two bits per LED), product of multiplication macro
R7 - Intermediary for generating the random sequence (used for determining which LED to turn on)
R8 - Contains the round's sequence generated by the MSP430 in the appropriate format.
R9 - Contains the number of LEDs used in a round
R10 - Intermediary for storing the user's input (copies P2IN)
R11 - Counter for how many rounds with the same amount of LEDs.
R12 - Contains the sequence entered by the user in the appropriate format.
R13 - If R13=0001h, the game is in No-LED mode (only sounds)
R14 - Delay for the sequence during the game
R15 - Random number stored in this register.