Question 1 (square room) :

```
#define F_CPU 1000000U

#include <avr/io.h>

#include <util/delay.h>

#include <stdlib.h>

#include <avr/interrupt.h>


#define ped 20


int interr = 0; //logic flag

int i=0;


int main(void)

{



        //initialize the ADC free-running mode

        ADC0.CTRLA |= ADC_RESSEL_10BIT_gc;

        ADC0.CTRLA |=ADC_FREERUN_bm;

        ADC0.CTRLA |= ADC_ENABLE_bm;

        ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc;

        //enable debug mode

        ADC0.DBGCTRL |= ADC_DBGRUN_bm;

        //Window comparator mode

        ADC0.WINLT |= 10;

        ADC0.INTCTRL |= ADC_WCMP_bm;

        ADC0.CTRLE |= ADC_WINCM0_bm;
```

```
while(i<3)
{



        PORTD_DIR |= PIN0_bm;                    //LED of right movement
        PORTD_DIR |= PIN1_bm;                    //LED of straight movement
        PORTD_DIR |= PIN2_bm;                    //LED of left movement

        //It goes straight ahead***********************************************

        PORTD_OUTCLR |= PIN1_bm;  //on the straight movement
        PORTD_OUT |= PIN0_bm;     //off right movement
        PORTD_OUT |= PIN2_bm;     //off left movement
        //breakpoint: we check the Leds and RES<10 so it goes in the ISR of the ADC

        //Makes the first left turn************************************************




        //interrupts for the ADC
        sei();
        ADC0.COMMAND |= ADC_STCONV_bm;  //breakpoint

        while (interr==0)
        {
        }

        interr = 0;
```

```
                TCA0.SINGLE.CNT = 0;
                TCA0.SINGLE.CTRLB = 0;
                TCA0.SINGLE.CMP0 = ped;
                TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV1024_gc;
                TCA0.SINGLE.CTRLA |= 1;
                TCA0.SINGLE.INTCTRL = TCA_SINGLE_CMP0_bm;
                sei();  //breakpoint

                while (interr==0)
                {
                }

                //It goes straight ahead*********************************************

                interr = 0;

                PORTD_OUT |= PIN2_bm;                   //LED of left movement is off
                PORTD_OUTCLR = PIN1_bm;                 //LED of straight movement is on
                // breakpoint:we check the Leds

                i++;
        }

        cli();
}

//for the ADC
```

```
ISR(ADC0_WCOMP_vect)

{

	int intflags = ADC0.INTFLAGS;

	ADC0.INTFLAGS = intflags;

	PORTD_OUTCLR = PIN2_bm;				//LED of left movement is on

	PORTD_OUT |= PIN1_bm;				//LED of straight movement is off

	interr = 1;				//breakpoint: we check the Leds


}


//for the timer

ISR(TCA0_CMP0_vect)

{

	TCA0_SINGLE_CTRLA = 0;

	int intflags = TCA0.SINGLE.INTFLAGS;

	TCA0.SINGLE.INTFLAGS = intflags;

	interr = 1;   //breakpoint

	//third breakpoint: we ckeck if it went in the ISR of the timer

}
```

Question 2-3 (different room and reverse function):

```
#define F_CPU 1000000U


#include <avr/io.h>

#include <util/delay.h>

#include <stdlib.h>
```

```
#include <avr/interrupt.h>

#define ped 20

int interr = 0;   //interrupt
int i, j  =0 ;  // repetitions until it reaches the starting position
int count_left = 0;  //left turns
int count_right = 0;  //right turns

int main(){

        PORTD_DIR |= PIN0_bm;                   //LED of right movement
        PORTD_DIR |= PIN1_bm;                   //LED of straight movement
        PORTD_DIR |= PIN2_bm;                   //LED of left movement

        while(i<7){   //7 corners

                PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
                sei(); //breakpoint

                PORTD_OUTCLR |= PIN1_bm;  //on the straight movement
                PORTD_OUT |= PIN0_bm;     //off right movement
                PORTD_OUT |= PIN2_bm;     //off left movement

                        while(interr==1){  //if I press switch, interr = 1 and it will get into here, where
the timer starts
                                TCA0.SINGLE.CNT = 0;
                                TCA0.SINGLE.CTRLB = 0;
                                TCA0.SINGLE.CMP0 = ped;
                                TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV1024_gc;
                                TCA0.SINGLE.CTRLA |= 1;
                                TCA0.SINGLE.INTCTRL = TCA_SINGLE_CMP0_bm;
                                sei();  //breakpoint
                                while(interr==1){   //when the timers ends, interr = 0
                                }

                                PORTD_OUT |= PIN0_bm;     //off right movement
                                PORTD_OUT |= PIN2_bm;     //off left movement

                                reverse();  //the function for the reverse path is enabled
                        }

                ADC0.CTRLA |= ADC_RESSEL_10BIT_gc;
                ADC0.CTRLA |=ADC_FREERUN_bm;
                ADC0.CTRLA |= ADC_ENABLE_bm;
                ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc;
                ADC0.DBGCTRL |= ADC_DBGRUN_bm;
                ADC0.WINLT |= 10;
                ADC0.WINHT |= 20;
                ADC0.INTCTRL |= ADC_WCMP_bm;
                ADC0.CTRLE |= ADC_WINCM_0_bm;
                ADC0.COMMAND |= ADC_STCONV_bm; //breakpoint, here you change result for the
wanted turn(<threshold=left, >threshold + STCONV bit = right)

                cli();  //breakpoint
        }
}
```

```
ISR(ADC0_WCOMP_vect)
{
        int intflags = ADC0.INTFLAGS; //breakpoint
        ADC0.INTFLAGS = intflags;
        if(ADC0.RES<10){   //result < threshold, for the left turn
                if(count_left<5){  //5 left turns in total
                        PORTD_OUT |= PIN1_bm; //straight path ends
                        PORTD_OUTCLR = PIN2_bm; //ready for left turn
                        _delay_ms(20);
                        PORTD_OUT |= PIN2_bm;  //breakpoint
                        PORTD_OUTCLR = PIN1_bm;
                        count_left++;
                        i++;  //1 turn is completed
                }
        }
        else if(ADC0.RES>20){   //result>threshold, for the right turn
                if(count_right<2){  //2 right turns in total
                        PORTD_OUT |= PIN1_bm; //straight path ends
                        PORTD_OUTCLR = PIN0_bm; //ready for right turn
                        _delay_ms(20);
                        PORTD_OUT |= PIN0_bm;  //breakpoint
                        PORTD_OUTCLR = PIN1_bm;
                        count_right++;
                        i++;
                }
        }
        else{}
}
```

```
ISR(TCA0_CMP0_vect){
        TCA0_SINGLE_CTRLA = 0;
        int intflags = TCA0.SINGLE.INTFLAGS;
        TCA0.SINGLE.INTFLAGS = intflags;  //breakpoint
        interr = 0;
        //all 3 leds are on as long as the device is turning 180 degrees
        PORTD_OUTCLR = PIN0_bm;
        PORTD_OUTCLR = PIN1_bm;
        PORTD_OUTCLR = PIN2_bm;
}



ISR(PORTF_PORT_vect){
        int intflags = PORTF.INTFLAGS;
        PORTF.INTFLAGS = intflags;
        interr = 1; //breakpoint
}


reverse(){   //reverse path function
        int sum = count_right + count_left;   //how many turns has it completed until now
        while(j<sum){ //here you execute the reverse path, before it gets into while loop, you change
result for the wanted turn and then it returns to get into while loop for j++
                j++;
                i = 7;
        }
}
```