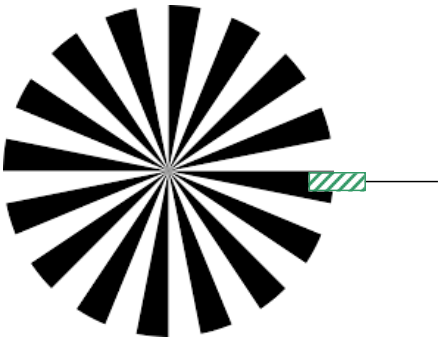
QUESTION 1 *Sensors*

Assume a sensor is connected to a rotating Siemens star (reflective and non-reflective segments), which will generate a rectangle signal depending on its rotational speed:

- Assume the sensor is connected to pin **1**
- Assume ports have already been initialized



(a) Polling

Complete the program below that continuously checks the sensors output and counts the encoder-ticks (falling edge) in global variable “ticks”.

```
// assume I/O line inits have been done
int ticks=0; // global variable for counting encoder ticks
int current, previous;

void loop()
{









}
```

(b) Interrupts

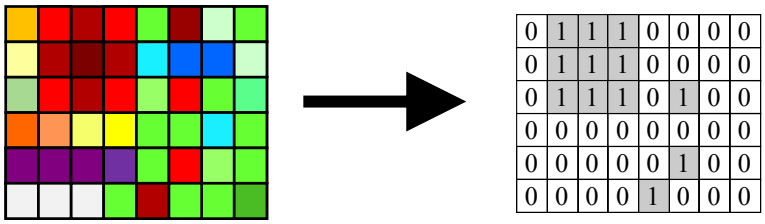
To achieve the same outcome, complete the interrupt service routine ISR below. You can assume that this routine will be called at every **falling edge of pin 1**.

Complete the interrupt subroutine ISR.

```
// assume I/O line and interrupt inits have been done
int ticks=0;

void ISR_encoder()
{




}
```

QUESTION 2 *COLOR IMAGES*

A color image has already been reduced to a binary image of matching values (`match`). Find the column *x* with the highest number of matching pixels.



image

0	1	1	1	0	0	0	0
0	1	1	1	0	0	0	0
0	1	1	1	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0

match

```
int find_column(BYTE match[160*120])
{ int result=0;









  return result;
}
```

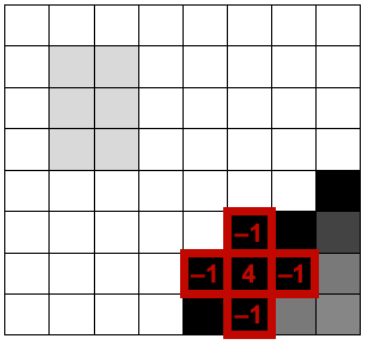
QUESTION 3 *GRAYSCALE IMAGES*

Filter templates can be used for many local image operations. These filters use the current pixel's value and that of its immediate neighbors.

The **Laplace filter** uses a 3x3 template to find grayscale changes or "edges" in an image, which very often indicate object outlines.

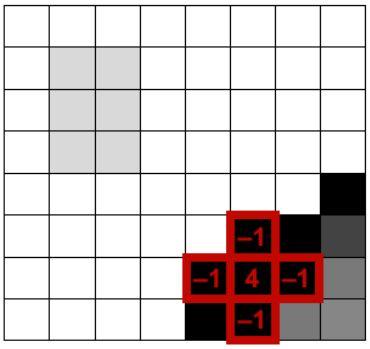
0	-1	0
-1	4	-1
0	-1	0

Example



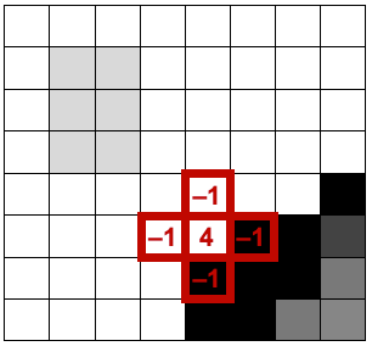
255	255	250	251	248	248	240	240
247	157	159	249	250	251	244	241
250	142	141	247	248	248	239	238
248	159	148	247	250	249	250	251
244	245	243	250	251	239	238	8
241	243	242	238	246	0	2	12
241	249	252	237	0	2	1	23
241	248	235	224	5	7	23	123

low absolute value



255	255	250	251	248	248	240	240
247	157	159	249	250	251	244	241
250	142	141	247	248	248	239	238
248	159	148	247	250	249	250	251
244	245	243	250	251	239	238	8
241	243	242	238	246	0	2	12
241	249	252	237	0	2	1	23
241	248	235	224	5	7	23	123

low absolute value



255	255	250	251	248	248	240	240
247	157	159	249	250	251	244	241
250	142	141	247	248	248	239	238
248	159	148	247	250	249	250	251
244	245	243	250	251	239	238	8
241	243	242	238	246	0	2	12
241	249	252	237	0	2	1	23
241	248	235	224	5	7	23	123

high absolute value

Write a program that implements the Laplace filter

```
int laplace(BYTE img[W*H], edge[W*H]) // Width * Height
{ int i,j, pos;

  // 1. Calculate edge for each inside pixel

  for (i= __; i< ___; i++)

    for (j= __; j< ___; j++)

    { pos = i*W + j;









    }

  // 2. Clear outside borders of 'edge'









}
```

QUESTION 4 *Data Transmission*

(a) Parallel Data transmission

To calculate the checksum, calculate the sum of all data bytes and only use the rightmost byte.

(a1) The following data bytes are to be transmitted:
0x12, 0xF0, 0x77, 0x25, 0x43
Calculate the single-byte sender checksum: _____

(a2) The following data bytes are being received:
0x12, 0xF0, 0x77, 0x23, 0x43, <check_sum>
Calculate the single-byte receiver checksum: _____

(b) Write a subroutine that transmits a sequence of bytes, then generates the checksum and transmits this as well.

- Use command '`Serial.write(myByte)`' for transmitting one byte at a time.
- Do **not** worry about synchronization issues; assume that every byte written to this port gets properly transmitted (i.e. non wait or handshake required).

```
void transmit(BYTE buffer[], int size)  //size = num. of bytes
{









}
```