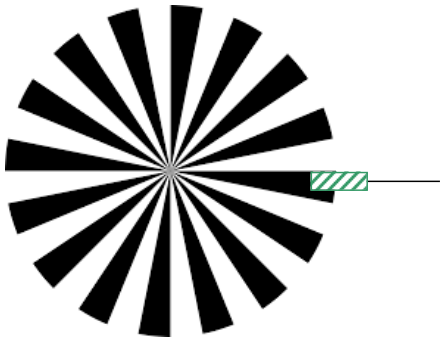
QUESTION 1 *Sensors*

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 **B0**
  (the other pins of port B are not used; you can assume they return 0)
- 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”.

```
int ticks; // global variable for counting encoder ticks

int main()
{ // assume port inits have been done

  int current, previous;

  ticks = 0;

  while(1)
  {









  }
}
```

(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 input B0.

Complete the interrupt subroutine ISR.

```
int ticks; // global variable for counting encoder ticks

int main()
{ // assume port inits have been done
  // enable interrupts

  // complete other tasks
}
```

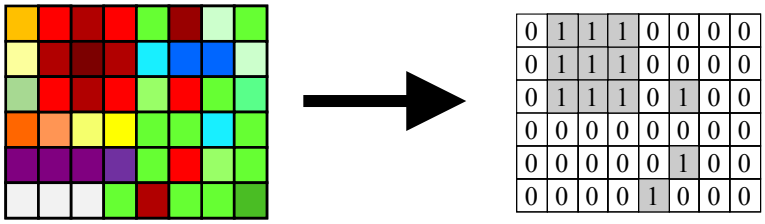
```
ISR(PCINT1_vect)
{



}
```

QUESTION 2 *DATA PROCESSING*

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 *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 port D for transmitting one byte at a time.
- Do **not** worry about synchronization issues; assume that every byte written to this port gets properly transmitted.

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









}
```