# Løsningsforslag

## Løsningsforslag Oppgave 1 (25%)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | b | a | b | c | a | c | c | b | d | d | b | b | c | a | d | b | c | b | c |

## Løsningsforslag Oppgave 2 (20%)

2a) LF: Linje 5, imid = (imin+imax)//2
2b) LF: [9,7,5,3,2,1], snur lista
2c) LF: 101011001, konverterer heltall til binærrepresentasjon
2d) LF: [2, 8, 4, 10, 7, 6, 3, 5, 9, 1] (tallene 1 til 10 i tilfeldig rekkefølge), randomiserer plasseringen av 10 tall i ei liste.

## Løsningsforslag Oppgave 3 (25%)

Kommentarer til løsninger:
3e) For å få full score måtte man ha en løsning som ikke vil kunne returnere duplikate reg.nr.
3f) For å få full score må løsningen ta hensyn til at rekkefølgen bilene passerte boks A og B kan være forskjellige.

3a) 6%

```
def file_to_table(filename):
  table = []
  f = open(filename,'r')
  for line in f:
    line = line.strip() # Remove white spaces
    data = line.split(',')
    for i in range(6): # Pick out date (3 digits) and time (3 digits)
      data[i]=int(data[i])
    table.append(data)
  f.close()
  return table
```

3b) 3%

```
# Not required function (should just be used in the solution):
import datetime

# THIS FUNCTION WAS NOT PART OF THE SOLUTION, BUT PROVIDED IN THE PROBLEM!
def date_diff(start,end):
  d0 = datetime.date(start[0],start[1],start[2])
  d1 = datetime.date(end[0],end[1],end[2])
  delta = d1-d0
  return delta.days

# THE SOLUTION:
def time_diff(start,end):
  secsPerDay = 60*60*24
  day_diff = date_diff(start[:3],end[:3])
  time_diff = ((end[5]+end[4]*60+end[3]*60*60)-
        (start[5]+start[4]*60+start[3]*60*60))
  return day_diff*secsPerDay+time_diff
```

```
# NOT VERY ACCURATE ALTERNATIVE SOLUTION (30.44 days in average/month)
def time_diff(start,end):
  start_sec=(start[5]+start[4]*60+start[3]*60*60+
             start[2]*60*60*24+start[1]*60*60*24*30.44+
             start[0]*60*60*24*30.44*12)
  end_sec=(end[5]+end[4]*60+end[3]*60*60+
             end[2]*60*60*24+end[1]*60*60*24*30.44+
             end[0]*60*60*24*30.44*12)
  return int(end_sec-start_sec)
```

3c) 5%

```
def check_min_distance(car_table,diff):
  crazy_drivers=[]
  for i in range(len(car_table)-1):
    first_car=car_table[i][0:6]
    sec_car=car_table[i+1][0:6]
    if time_diff(first_car,sec_car)<diff:
      crazy_drivers.append(car_table[i+1][6])
  return crazy_drivers
```

3d) 4%

```
def list_el_cars(car_table):
  el_cars=0
  for item in car_table:
    plate= item[6]
    if plate[:2]=='EK' or plate[:2]=='EL' or plate[:2]=='EV':
      el_cars+=1
  return el_cars
```

3e) 5%

```
import random

def generate_license_numbers(amount):
  letters=('BS','CV','EL','FY','KU','LE','NB','PC','SY','WC')
  numbers = []
  for x in range(amount):
    plate=0
    while plate not in numbers:
      plate=random.choice(letters)+str(random.randint(10000,99999))
      numbers.append(plate)
  return numbers
```

3f) 7%

```
def list_speeders(filename_a,filename_b,speed_limit,distance):
  time_limit = (distance/speed_limit)*3600
  speeders = []
  A=file_to_table(filename_a)
  B=file_to_table(filename_b)
  for item_a in A:
    for item_b in B:
      if item_a[6]==item_b[6]: # Same numberplate
        sec = time_diff(item_a[:6],item_b[:6])
        if sec<time_limit:
          speeders.append(item_a[6])
  return speeders
```

## Løsningsforslag Oppgave 4 (30%)

4a) (3%)

```
def formatTime(seconds):
    hours = seconds//3600
    mins = (seconds % 3600)//60
    secs = seconds % 60
    if hours < 10:
        hh = "0" + str(hours)
    else:
        hh = str(hours)
    if mins < 10:
        mm = "0" + str(mins)
    else:
        mm = str(mins)
    if secs < 10:
        ss = "0" + str(secs)
    else:
        ss = str(secs)
    return hh + ":" + mm + ":" + ss
```

**En litt mer elegant løsning:**

```
def formater(time):
    if time < 10:
        return '0'+str(time)
    else:
        return str(time)

def formatTime(seconds):
    hours = seconds//3600
    mins = (seconds%3600)//60
    secs = seconds%60
    streng = formater(hours)+':'+formater(mins)+':'+formater(secs)
    return streng
```

4b) (2%)

```
def valuesDecember():
     first = 3*3600 +18*60 # 3:18 December 1st in seconds
     period = 12*3600+25*60+12 # 12:25:12 in seconds
     return first, period
```

4c) (5%)

```
def genTides():
     lows = []
     highs = []
     start, period = valuesDecember()
     secPerMonth = 24*60*60*31 # 2678400 secs (31 days,24 hours,60 min in sec)
     tide = start
     while tide<secPerMonth:
          lows.append(tide)
          highs.append(tide+period//2)
          tide+= period
     return lows,highs
```

4d) (3%)

```
def genTidesStr(tideList):
     formatedList= []
     secPerDay = 24*60*60 # 86400 seconds
     for item in tideList:
          day = (item//secPerDay)+1
          time = item-day*(24*60*60)
          formatedList.append(str(day)+ " "+str(formatTime(item % secPerDay)))
     return formatedList
```

4e (7%)

```
def checkTides(dayInMonth):
     lows,highs = genTides()
     secPerDay  = 24*60*60 # 86400 seconds
     start_time = dayInMonth*secPerDay+9*(60*60) # 09:00 at dayInMonth
     end_time   = start_time+4*(60*60) # 4 hours after 9:00 (13:00)

     for item in lows:
          if start_time <= item <= end_time:
               print('low tide at',formatTime(item % secPerDay))
               return
     for item in highs:
          if start_time <= item <= end_time:
               print('high tide at',formatTime(item % secPerDay))
               return
     print('no tides')
```

4f (5%)

```
def listTides():
     lows, highs = genTides()
     secPerDay = 24*60*60 # 86400 seconds
     i = 0
     print('Day'.rjust(3),'First'.center(8),'Second'.center(8))

     for day in range(1,32): # days from 1 to 31 (including 31)
          line =str(day).rjust(3)
          while (i < len(lows)) and (lows[i] < day*secPerDay):
               line += ' '+ str(formatTime(lows[i] % secPerDay))
               i += 1
          print(line)
```