1. (4 Punkte) Implementiere die Funktion.

```
def mehrKleine(s):
    '''
    s: String
    returns: True, wenn in s mehr Zeichen a–z als Zeichen A–Z vorkommen
    '''
```

Lösung:

```
def mehrKleine(s):
    summeKlein = 0
    summeGross = 0
    for c in s:
        if 'a' <= c <= 'z': summeKlein += 1
        elif 'A' <= c <= 'Z': summeGross += 1
    return summeKlein > summeGross
```

2. (3 Punkte) Implementiere die Funktion.

```
def anzahlZiffern(s):
    '''
    s: String
    returns: int, Anzahl der Ziffern in s
    '''
```

Lösung:

```
def anzahlZiffern(s):
    zaehl = 0
    for c in s:
        if '0' <= c <= '9': zaehl += 1
    return zaehl
```

3. (3 Punkte) Implementiere die Funktion. Die eingebauten Python-Funktionen min und max dürfen nicht verwendet werden.

```
def minZahl(a,b,c,d,e):
    '''
    a,b,c,d,e: Zahlen (ints oder floats)
    returns: das Minimum der Zahlen.
    '''
```

Lösung:

```
def minZahl(a,b,c,d,e):
    king = a
    if b < king: king = b
    if c < king: king = c
    if d < king: king = d
    if e < king: king = e
    return king
```

4. (5 Punkte) Die Quersumme einer Zahl ist die Summe ihrer Ziffern. Implementiere die Funktion maxQuer.

```
def maxQuer(a,b,c,d):
    '''
    a,b,c,d: positive ganze Zahlen
    returns: die Zahl mit der größten Quersumme.
    '''
```

Lösung:

```
def quersumme(k):
    summe = 0
    for c in str(k):
        summe += int(c)
    return summe

def maxQuer(a,b,c,d):
    '''
    a,b,c,d: positive ganze Zahlen
    returns: die Zahl mit der größten Quersumme.
    '''
    king = a
    kingWert = quersumme(a)
    if quersumme(b) > kingWert:
        kingWert = quersumme(b)
        king = b
    if quersumme(c) > kingWert:
        kingWert = quersumme(c)
        king = c
    if quersumme(d) > kingWert:
        kingWert = quersumme(d)
        king = d
    return king
```

5. (5 Punkte) Implementiere die Funktion.

```
def groessterRest(s):
    '''
    s: String mit Ziffern, mindestens Länge 2
    returns: int, Index an dem der zweistellige Teilstring beginnt
        der den größten Rest bei der Division durch 7 ergibt.

    Beispiel:
    >>> groessterRest('532343456')
    3
    denn bei Index 3 beginnt die zweistellige Zahl 34,
```

```
    die bei Division durch 7 den Rest 6 hat.
'''
```

Lösung:

```
def groessterRest(s):
    king = 0
    kingRest = int(s[:2]) % 7

    for i in range(1,len(s)-1):
        teil = int(s[i:i+2])
        if teil % 7 > kingRest:
            king = i
            kingRest = teil % 7
    return king
```