2 Funktionale Programmierung

1.1 Beziehen Sie sich bei ihren nachfolgenden Erläuterungen auf die entsprechenden Codezeilen. **Erläutern Sie**, ob das nachfolgende Codesegment die Kriterien der funktionalen Programmierung erfüllt.

```
1  L=[1,2,3,4,5,6,7,8,9,10]
2  for i in range(len(L)):
3      L[i]=L[i]**2
4  for i in L:
5      if i%2!=0:
6          L.remove(i)
7  summe = 5
8  for i in L:
9      summe += i
10 print(summe)
```

Lösung:

Die Zuweisungen in den Zeilen 1 und 7, der print Aufruf in Zeile 10, der Aufruf von *remove* in Zeile 6 sowie die bedingte Verzweigung in Zeile 5 stellen Funktionsaufrufe dar, sodass keine Anpassung notwendig ist. Die Zuweisungen in Zeile 3 sowie 9 entsprechen nicht einer funktionalen Realisierung, weil eine Berechnung ohne einen Funktionsaufruf durchgeführt wird. Die drei Zählschleifen in den Zeilen 2,4,8 müssen für ein funktionales Programm ebenfalls angepasst werden, da die Iteration kein Konzept der funktionalen Programmierung darstellt.

```
1  #ZW = Zuweisung
2  L=[1,2,3,4,5,6,7,8,9,10] #ZW ohne Berechnung
3  for i in range(len(L)): #Zählschleife
4      L[i]=L[i]**2        #ZW mit Berechnung
5  for i in L:             #Zählschleife
6      if i%2!=0:          #Funktionsaufruf
7          L.remove(i)     #Funktionsaufruf
8  summe = 5               #ZW ohne Berechnung
9  for i in L:             #Zählschleife
10     summe += i          #ZW mit Berechnung
11 print(summe)            #Funktionsaufruf
```

1.2 Nutzen Sie bei dieser Aufgabe die drei Funktionen *filter*, *map* und *reduce*. **Passen Sie** den Code aus 1.1 **an**, sodass er die Kriterien der funktionalen Programmierung erfüllt.

Lösung:

```
1 from functools import reduce
2 L=[1,2,3,4,5,6,7,8,9,10]
3 L=list(map(lambda x: x**2,L))
4 L=list(filter(lambda x: x%2==0,L))
5 summe = reduce(lambda x,y: x+y,L,5)
6 print(summe)
```

2. **Implementieren Sie** den nachfolgenden Code rekursiv.
Hinweis: Die Datenstruktur liste müssen Sie nicht bei der rekursiven Implementierung berücksichtigen.

```
1  def first(xs):
2      return xs[0]
3
4  def rest(xs):
5      return xs[1]
6
7  def map(xs, func):
8      if(xs==()):
9          return xs
10     liste = []
11     while rest(xs)!=():
12         liste.append(func(first(xs)))
13         xs=rest(xs)
14     liste.append(func(first(xs)))
15     xsNew=()
16     for i in liste[::-1]:
17         xsNew =(i,xsNew)
18     return xsNew
19
20 xs=(3,(5,(8,())))
21 xs=map(xs, lambda x: x**2)
22 print(xs)
```

Lösung:

```
1  def first(xs):
2      return xs[0]
3
4  def rest(xs):
5      return xs[1]
6
7  def map(xs, func):
8      if(xs==()):
9          return xs
10     if rest(xs)==():
11         return (func(first(xs)),())
12     return (func(first(xs)),map(rest(xs),func))
13
14 xs=(3,(5,(8,())))
15 xs=map(xs, lambda x: x**2)
16 print(xs)
```