Übung 6 – Musterlösung

Aufgabe 1: Schreiben Sie eine Funktion reverse[T](l:List[T]):List[T] die eine beliebige Liste unter der Verwendung von foldLeft oder foldRight umdreht.

```
def reverse[T](l:List[T]):List[T] = l.foldLeft(List():List[T]) ((a,b)=> b::a)
```

Aufgabe 2: Schreiben Sie eine Funktion *foldLeft[B,T] (base :B, l:List[T]) (B,T)=>b)* unter Verwendung von foldRight.

```
def foldRightViaFoldLeft[A,B](l: List[A], z: B)(f: (A,B) => B): B =
        l.reverse.foldLeft(z)((b,a) => f(a,b))
```

Aufgabe 3: Gegeben sei die folgende Liste, die ausdrückt, welche Programmiersprachen, welche Programmiersprachen welche Paradigmen unterstützen:

val Paradigmen=List(("erlang", "funktional"), ("erlang", "logisch"), ("prolog", "logisch"), ("scala", "funktional"), ("scala", "objektorientiert"), ("scapla", "logisch"), ("java","objektorientiert"))

a) Ermitteln Sie unter Verwendung von foldLeft oder foldRight welches Paradigma wie häufig in der Liste vorkommt wurde.

```
val p= Paradigmen.foldLeft(Map():Map[String,Int])((a,b)=>
a.updated(b._2,a.getOrElse(b._1,0)+1))
```

b) Berechnen Sie aus dem Ergebnis von a die relative Häufigkeit, mit der ein Paradigma vorkommt.

```
p.mapValues(x=>x.toDouble/Paradigmen.size)
```

Aufgabe 4: Gegeben sei die Funktion mapReduce mit dem folgenden Implementierung:

```
def mapReduce[S,B,R](mapFun:(S=>B), redFun:(R,B)=>R, base:R, l:List[S]):R =
    l.map(mapFun).foldLeft[R](base)(redFun)
```

Berechnen Sie mit dieser Funktion für alle Werte der Eingabeliste die kleinsten Primteiler und addieren Sie diese.

```
def findSmallestPrimeDivisor(l:Int, counter:Int):Int= l match {

    case _ if (l<1) => throw new Error("Negative Number of Zero")
    case 1 => 1
    case _ if (l % counter == 0) => counter
    case _ => findSmallestPrimeDivisor(l, counter+1)

 }

mapReduce((x:Int)=>findSmallestPrimeDivisor(x,2), (x:Int,y:Int)=>x+y,0,l)
```

Aufgabe 5: Schreiben Sie eine Funktion partial mit der folgenden Signatur:
def partial[A,B,C](a:A,f:(A,B)=>C):B=>C
Die Funktion bekommt als Parameter:

- eine Funktion mit 2 Variablen sowie
- ein Wert mit der die Funktion belegt werden soll.

Sie soll eine Funktion zurück liefern, bei der erste Parameter bereits belegt wurde.

Beispiel: partial(1, (a,b)=>a+b) soll eine Funktion zurückliefern, die einen Wert um 1 erhöht.

def partial[A,B,C](a:A, f:(A,B)=>C):B=>C = (b:B)=> f(a,b)