Aufgabe 1

1. **map** ((u,v),w){

return (v,m)

}

**shuffle** (TupelList ((v1,m1), …,(vn,mn)) ){

define (NewTupelList)

define (M)

sort (TupelList)

for ( i = unique values of v){

M = NULL

for ( j =0 ; length (TupelList)-1;j++) {

if (vj == i)

M.add(mj)

}

NewTupelList.add( (i,M)

}

return (NewTupelList)

}

**reduce** (v,M){

return (v,sum(M))

}

Aufgabe 2

1. Records: Sind die typischen Tupel aus Key Value Paaren, die aus den Splits generiert werden.

Splits: Teile an Daten. Die Eingabedaten werden bevor sie gemapped werden in Splits aufgeteilt. Diese können dann auf die Systeme verteilt werden.

1. Als erstes werden die Daten in Splits aufgeteilt, die an verschieden Mapper verteilt werden können. Danach wird aus jedem Split Records erstellt, die dann vom Mapper verarbeitet werden können.

InputFormat: Aufteilen der Input Daten in basierend auf der Größe des Input files in byte.

Inputsplit: Aufgeteilter Teil der Daten, die durch einen zugewiesenen Mapper verarbeitet werden.

RecordReader: Erstellt aus den Input Daten Key Value Paare, die vom Mapper verarbeitbar sind. Wird auch verwendet um die Grenzen der Splits festzulegen, damit keine Daten verloren gehen.

1. Die Eingabedaten sind eine Liste aus Records (Key, Value) die nach Splitten und lesen bzw. finden von map verarbeitet werden können.
2. Cleanup: Wird am Ende eines runs aufgerufen. Macht nichts, kann aber dementsprechend noch bearbeitet werden.
3. Setup: Wird am Anfang runs aufgerufen. Macht nichts, kann aber dementsprechend noch bearbeitet werden.

Run: Als erstes wird Setup aufgerufen. Für jedes übergebene Tupel ruft die Methode map() auf. Sie kann auch noch erweitert werden. Nach durchlaufen aller Eingaben wird cleanup(context) aufgerufen.