W2D1. Solution A: Calculate relative frequency by Pairs approach.

INPUT	**Input Split-1**	**Input Split-2**

Mapper Input	15 91 80 12 19 80
	17 15 80 18 91 18
	
	Neighbours:
	
	N(15) = {91,80,12,19,80}
	N(91) = {80,12,19,80}
	N(80) = {12,19,80}
	N(12) = {19,80}
	N(19) = {80}
	N(80) = { }
	
	N(17) = {15,80,18,91,18}
	N(15) = {80,18,91,18}
	N(80) = {18,91,18}
	N(18) = {91}
	N(91) = {18}
	N(18) = { }
		19 15 80 18 91 18
		18 15 18 18 80 18
		
		Neighbours:
		
		N(19) = {15,80,18,91,18}
		N(15) = {80,18,91,18}
		N(80) = {18,91,18}
		N(18) = {91}
		N(91) = {18}
		N(18) = { }
		
		N(18) = {15}
		N(15) = {18,18,80,18}
		N(18) = { }
		N(18) = {80}
		N(80) = {18}
		N(18) = { }

MAP	**Mapper-1**	**Mapper-2**
Mapper Output	((15,91),1) ((15, 0),1) ((15,80),1) ((15, 0),1) ((15,12),1) ((15, 0),1) ((15,19),1) ((15, 0),1) ((15,80),1) ((15, 0),1) ((91,80),1) ((91, 0),1) ((91,12),1) ((91, 0),1) ((91,19),1) ((91, 0),1) ((91,80),1) ((91, 0),1) ((80,12),1) ((80, 0),1) ((80,19),1)	((19,15),1) ((19, 0),1) ((19,80),1) ((19, 0),1) ((19,18),1) ((19, 0),1) ((19,91),1) ((19, 0),1) ((19,18),1) ((19, 0),1) ((15,80),1) ((15, 0),1) ((15,18),1) ((15, 0),1) ((15,91),1) ((15, 0),1) ((15,18),1) ((15, 0),1) ((80,18),1) ((80, 0),1) ((80,91),1)

	((80, 0),1) ((80,80),1) ((80, 0),1) ((12,19),1) ((12, 0),1) ((12,80),1) ((12, 0),1) ((19,80),1) ((19, 0),1) ((17,15),1) ((17, 0),1) ((17,80),1) ((17, 0),1) ((17,18),1) ((17, 0),1) ((17,91),1) ((17, 0),1) ((17,18),1) ((17, 0),1) ((15,80),1) ((15, 0),1) ((15,18),1) ((15, 0),1) ((15,91),1) ((15, 0),1) ((15,18),1) ((15, 0),1) ((80,18),1) ((80, 0),1) ((80,91),1) ((80, 0),1) ((80,18),1) ((80, 0),1) ((18,91),1) ((18, 0),1) ((91,18),1) ((91, 0),1)	((80, 0),1) ((80,18),1) ((80, 0),1) ((18,91),1) ((18, 0),1) ((91,18),1) ((91, 0),1) ((18,15),1) ((18, 0),1) ((15,18),1) ((15, 0),1) ((15,18),1) ((15, 0),1) ((15,80),1) ((15, 0),1) ((15,18),1) ((15, 0),1) ((18,80),1) ((18, 0),1) ((80,18),1) ((80, 0),1)
SHUFFLE & SORT		
Reducer Input	((12, 0), [2]) ((12,19), [1]) ((12,80), [1]) ((15, 0), [17]) ((15,12), [1]) ((15,18), [1,1,1,1,1,1,1]) ((15,19), [1]) ((15,80), [1,1,1,1,1]) ((15,91), [1,1,1])	

Sorting rule:
class Pair implements
Comparable<Pair> {
 int a, b;
 int compareTo(Pair p) {
   int k = a.compareTo(p.a)
   if(k==0) k=b.compareTo(p.b)
   return k;
 }
}
	((17, 0), [5])
	((17,15), [1])
	((17,18), [1,1])
	((17,80), [1])
	((17,91), [1])
	
	((18, 0), [4])
	((18,15), [1])
	((18,80), [1])
	((18,91), [1,1])
	
	((19, 0), [6])
	((19,15), [1])
	((19,18), [1,1])
	((19,80), [1,1])
	((19,91), [1])
	
	((80, 0), [10])
	((80,12), [1])
	((80,18), [1,1,1,1,1])
	((80,19), [1])
	((80,80), [1])
	((80,91), [1,1])
	
	((91, 0), [6])
	((91,12), [1])
	((91,18), [1,1])
	((91,19), [1])
	((91,80), [1,1])	

REDUCE	**Reducer-1**	
Reducer Output	((12,19), 1 / 2) ((12,80), 1 / 2) ((15,12), 1 / 17) ((15,18), 7 / 17) ((15,19), 1 / 17) ((15,80), 5 / 17) ((15,91), 3 / 17) ((17,15), 1 / 5) ((17,18), 2 / 5) ((17,80), 1 / 5) ((17,91), 1 / 5) ((18,15), 1 / 4) ((18,80), 1 / 4) ((18,91), 2 / 4)	

	((19,15), 1 / 6) ((19,18), 2 / 6) ((19,80), 2 / 6) ((19,91), 1 / 6) ((80,12), 1 / 10) ((80,18), 5 / 10) ((80,19), 1 / 10) ((80,80), 1 / 10) ((80,91), 2 / 10) ((91,12), 1 / 6) ((91,18), 2 / 6) ((91,19), 1 / 6) ((91,80), 2 / 6)