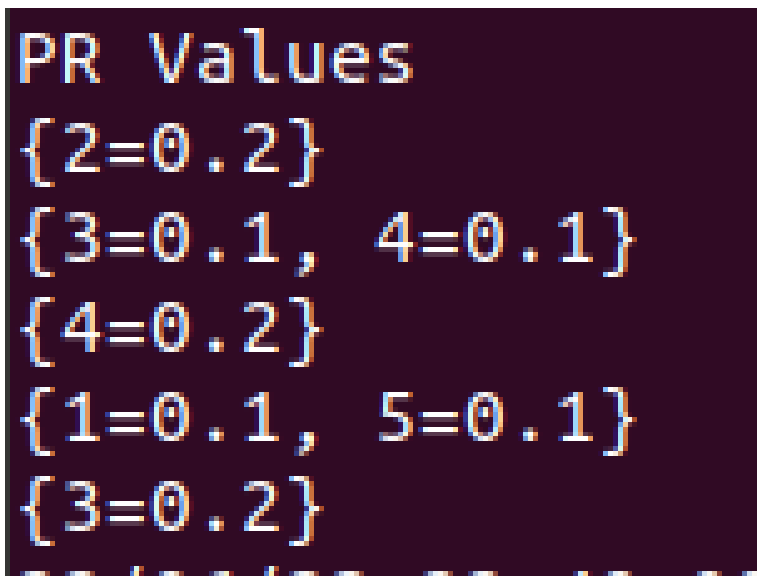
HW2

1. Explanation of the source code
    a. How is the Mapper function defined? Which kind of intermediate results are generated?
        1. The mapper function’s purpose is to get the data store it into a hashmap and give the following output in context.write()




```
public void map(Object key, Text value, Context context
                   ) throws IOException, InterruptedException {
      // You need to complete this function.
    // store the mapped function into contaxt
      //      context.write(key, value);
      String val = value.toString();
      HashMap<Integer, Double> hash_map = new HashMap<Integer, Double>();
      ArrayList<Double> listOfValues =
      vPRValues.values().stream().collect(Collectors.toCollection(ArrayList::new));
      char [] temp = new char[val.length()];
      int [] intValues = new int[val.length()];




      for(int i = 0; i < val.length(); i++) {
          if(val.charAt(i) != ' ') {
            if (i != 0 && val.length()-3 == 0) {
              try {
                hash_map.put(Character.getNumericValue(val.charAt(i)),
                listOfValues.get(i));
              }catch(Exception e){
//                System.out.println(i + " = Error");
              }
            }else if(i != 0 && val.length()-3 > 1) {
              try {
                hash_map.put(Character.getNumericValue(val.charAt(i)),
                listOfValues.get(i)/(val.length()-3));
              }catch(Exception e){
//                System.out.println(i + " = Error");
              }
             }

          }
      }
//      DoubleWritable valueSet_text =
  //new DoubleWritable(Integer.parseInt(hash_map.values().toString()));
      Object [] keyString =  hash_map.keySet().toArray();
      Object [] valueString =  hash_map.values().toArray();
      IntWritable keySet_text = new IntWritable((int)keyString[0]);
      for(int i = 0; i < valueString.length; i++) {
        context.write(keySet_text, new DoubleWritable((double)valueString[i]));
      }
      System.out.println(hash_map);
    }
  }
```

```
public void reduce(IntWritable key, Iterable<DoubleWritable> values, Context context) throws IOException, InterruptedExcepti
on
    {
      double [] arr = new double[4];
```

```
      double num = 0;
      for(DoubleWritable val : values) {
        num += val.get();
      }
      double finalVal = (0.85 * num) + ((1 - 0.85)/5);
      num = finalVal;
      context.write(key, new DoubleWritable(num));


    }
  }
```

a. How is the Reducer function defined? How do you aggregate the intermediate results and get the final outputs?

b. The reduce function takes the data from the mapper function and compresses the data and computes the PageRank. The computation is done using the following formula:

   a. double finalVal = (0.85 * num) + ((1 - 0.85)/5);

      Final val the output.

      num is the number that is retrieved after computation the following:

      1 0.1

      2 0.2

      3 0.3

      4 0.3

      5 0.2

```
public void reduce(IntWritable key, Iterable<DoubleWritable> values, Context context) throws IOException, InterruptedException
    {
      double [] arr = new double[4];
      double num = 0;
      for(DoubleWritable val : values) {
        num += val.get();
      }
      double finalVal = (0.85 * num) + ((1 - 0.85)/5);
      num = finalVal;
      context.write(key, new DoubleWritable(num));


    }
```

1. Do you use a Combiner function? Why or why not?

   a. No. I did not use a combiner function, because hadoop automatically will sort and compress the data.

2.1 Output.

```
1 1	183.25391226918666
2 2	366.4778245383733
3 3	549.70173680756
4 4	366.4778245383733
```

2.2 The output is not close to the close the actual page rank. The mapping has given out the correct response. The computation is also done correctly.

The code has a bug some type of power function is not yet discerned for the large difference in output.