Q1[4]. Consider a Map/Reduce solution to a program that reads a file full with integers and computes the number of integers that are divisible by 3.

1. What is the input/output type of the mapper and reducer?

|  |
| --- |
| Mapper<LongWritable, Text, Text, IntWritable>  Reducer<Text, IntWritable, Text, IntWritable> |

1. Show the map method for the program.

|  |
| --- |
| public void map(LongWritable key, Text value, Context context)  throws IOException, InterruptedException {  THE\_LOGGER.debug("I AM IN LOGGER");  String valueAsString = value.toString().trim();  if (Integer.parseInt(valueAsString) % 3 == 0) {  context.write(new Text("1"), new IntWritable(Integer.parseInt(valueAsString)));  } else {  context.write(new Text("0"), new IntWritable(Integer.parseInt(valueAsString)));  }  } |

Note we preserve the count of numbers not divisible by three here too. This is just in case we want to check the ratio, there is no reason other than that.

1. Show the reduce method for the program.

|  |
| --- |
| public void reduce(Text isDivByThree,  Iterable<IntWritable> count, Context context)  throws IOException, InterruptedException {  double sum=0;  int c = 0;  for(IntWritable el: count){  c += 1;  }  context.write(isDivByThree, new IntWritable(c));  } |

1. Can you create a combiner for the program? What will be the code for the reduce method of the combiner? Do you need to change the map or reduce method to support the combiner? If so, show the new map and reduce method.

|  |
| --- |
| public void reduce(Text isDivByThree,  Iterable<IntWritable> count, Context context)  throws IOException, InterruptedException {  double sum=0;  int c = 0;  for(IntWritable el: count){  c += 1;  }  context.write(isDivByThree, new IntWritable(c));  } |

The combiner is the same as the reducer. There is no reason to add a combiner here, so the map and reduce methods are unchanged.

Q2[4]. Consider a Map/Reduce solution to a program that reads a file that contains the date and a temperature that is measured for this day. There can be multiple temperatures that are measured each day. The program should calculate the highest temperature for each day.

1. What is the input/output type of the mapper and reducer?

|  |
| --- |
| Mapper<LongWritable, Text, Text, IntWritable>  Reducer<Text, IntWritable, Text, DoubleWritable> |

1. Show the map method for the program.

|  |
| --- |
| public void map(LongWritable key, Text value, Context context)  throws IOException, InterruptedException {  THE\_LOGGER.debug("I AM IN LOGGER");  String valueAsString = value.toString().trim();  String[] tokens = valueAsString.split(" ");  if (tokens.length != 2) {  return;  }  context.write(new Text(tokens[0]), new IntWritable(Integer.parseInt(tokens[1])));  //single write, but we can have multiple writes to context  } |

1. Show the reduce method for the program.

|  |
| --- |
| @Override  public void reduce(Text date,  Iterable<IntWritable> temperatures, Context context)  throws IOException, InterruptedException {  double max=-1;  int count = 0;  for(IntWritable el: temperatures){  if (el.get() > max) {  max = el.get();  }  }  context.write(date, new DoubleWritable(max));  } |

Note: here we assume temp >= 0. It’s simpler for our small program and could be changed easily.

1. Can you create a combiner for the program? What will be the code for the reduce method of the combiner? Do you need to change the map or reduce method to support the combiner? If so, show the new map and reduce method.

The code for the reduce method of the combiner would be the same as the reduce method of the reducer. No code would need to be changed. We would not need to change the map or reduce method to support a combiner.