Problem 1

(a) Please refer to files the SVN repository
(b) Please refer to files the SVN repository
(c) The last seven lines of the output on the input file nameyeartestdata

(Smith,John) 3

(Turing,Alan) 1

(Wamsley,Jayme) 1

(Webre,Josh) 1

(Weston,Clark) 1

(Woodburn,Louis) 1

(Woodburn,Providencia) 1

Problem 2

(a) It is very hard to parallelize unsorted grouping because unsorted data input will be distributed across different work nodes and merging groups from different mappers requires the comparison of the current group with all other groups. Additionally, there is no partial sorting. Groups are not ordered and the elements inside any group are not ordered either. On the contrary, sorted grouping can be easily parallelized. Because of the total order, grouping will only require the comparison of the current group with previous group to perform group merge if there is any.
When the map reduce job fails, it is because it will sorting use extra space. Fast sorting algorithm such as quick sort needs $O(\log n)$ extra space and merge sort require $O(n)$ extra space. In place sorting such as bubble sort is very slow and hence is not considered. So in the cases where the input values fit into the working memory, the sorting process will require extra memory and hence fail the map reduce job.

(b) It is not globally sorted because there is no order between the reducers. NameYearPartioner guarantees that same last name will end up in the same reducer but does not provide any other constraints that would result in reducers are totally ordered.
The youngest person with last name Newton is born on 1987. The complete output is
Newton Candace 1987-Sep-05

(c) Please refer to the files in the SVN repository

(d) The job is configured to run WITHOUT the customized sort comparator. By default, the data is sorted by both name and year ascending. Therefore we will have the output be the oldest person instead of the youngest person.

(e) Because the data being passed to the reduce method is being grouped according to the full key (name and year), so multiple records with the same last name (but different years) are being output. We want it to be grouped by name only so and the way to achieve this is to customize the group comparator that groups the data based only on the last name.

Problem 3

(a) The input format that can be used to read the data directly as key value pair is KeyValueTextInputFormat. Files are broken into lines and each line is divided into key and value parts by a separator byte, in this case it will be divided by the tab character.
The code to retrieve the filename is typed pasted below. We need to import org.apache.hadoop.mapreduce.lib.input.FileSplit;

```
FileSplit fsplit = (FileSplit) context.getInputSplit();
   Path path = fsplit.getPath();
   String fileName = path.getName();
```

(b) Please refer to the files in the SVN repository
(c) The output for the Hamlet excerpt is

have Hamlet@282

heaven Hamlet@282

and Hamlet@282

earth Hamlet@282

there Hamlet@133

are Hamlet@133

more Hamlet@133

things Hamlet@133

in Hamlet@133

heaven Hamlet@133

and Hamlet@133

earth Hamlet@133