Names: Bob Skowron, Jason Walker
Keys: rskowron, jwalker
SVN: jwalker: https://svn.seas.wustl.edu/repositories/jwalker/cse427s_fl17/

1. a. The *InputFormat* to use is KeyValueTextInputFormat. The default split is tab so no extra settings necessary.
   Code to add to driver: *gob.setInputFormatClass(KeyValueTextInputFormat.class*
   To get the file path, we use the context to get the current input split and then read the path and name
   FileSplit fileSplit = (FileSplit)context.getInputSplit();
   String filename = fileSplit.getPath().getName();

   b. See SVN

   c. Mapper output for lines from Hamlet:
   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

Names: Bob Skowron, Jason Walker
Keys: rskowron, jwalker
SVN: jwalker: https://svn.seas.wustl.edu/repositories/jwalker/cse427s_fl17/

2. a. Cosine similarity: $\frac{r_x^T \cdot r_y}{\|r_x\| \cdot \|r_y\|}$

   This is equivalent to: $\frac{\sum_{s \epsilon S_{xy}} r_{xs} * r_{ys}}{\sqrt{\sum_{s \epsilon S_{xy}} r_{xs}^2} * \sqrt{\sum_{s \epsilon S_{xy}} r_{ys}^2}}$

   Normalizing (i.e. subtracting the relevant means) gives: $\frac{\sum_{s \epsilon S_{xy}} (r_{xs} - \bar{r_x}) * (r_{ys} - \bar{r_y})}{\sqrt{\sum_{s \epsilon S_{xy}} (r_{xs} - \bar{r_x})^2} * \sqrt{\sum_{s \epsilon S_{xy}} (r_{ys} - \bar{r_y})^2}}$

   Which is the Pearson correlation.

   b.
   - An advantage of normalization is to remove bias. We can remove users who are overly critical (all low scores) or overly enthusiastic (all high rankings)
   - With Pearson correlation you need to compute and store the average rating for each user. This requires us to process the entire set of ratings.

   c. One disadvantage of the Jaccard similarity is that it does not take into account the value of the rating. Thus, even if two users rate the exact same items with opposite ratings it would have a high Jaccard similarity ranking. To overcome this problem, you could initially group by rating and then perform the similarity measure.

Names: Bob Skowron, Jason Walker
Keys: rskowron, jwalker
SVN: jwalker: https://svn.seas.wustl.edu/repositories/jwalker/cse427s_fl17/

3. a. The dual approach can be more efficient if the #users $\gg$ #items. Also, it can be easier to find similar items.

b.

User	Movie1	Movie2	Movie3	Movie5
user1	1	3	2	
user2		2	3	5
user3	1	2		

Mapper Output (pairs of movies and ratings):
((movie1,movie2), (1,3))
((movie1,movie3), (1,2))
((movie2,movie3), (3,2))
((movie2,movie3), (2,3))
((movie3,movie5), (3,5))
((movie1,movie2), (1,2))

Reducer Input:
((movie1,movie2), [(1,3),(1,2)])
((movie1,movie3), (1,2))
((movie2,movie3), [(3,2),(2,3)])
((movie3,movie5), (3,5))
((movie1,movie2), (1,2))

Reducer Output (similarity measure of movie pairs):
((movie1,movie2), .99)
((movie1,movie3), 0)
((movie1,movie5), 0)
((movie2,movie3), .92)
((movie2,movie5), 0)
((movie3,movie5), 0)