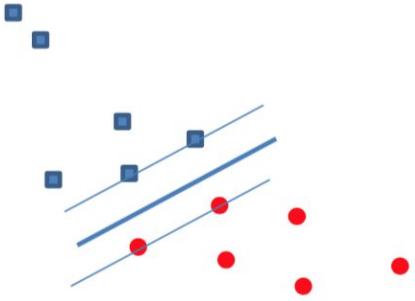
SENG 474 A3

1. (a)



(b)
Distance is computed by:

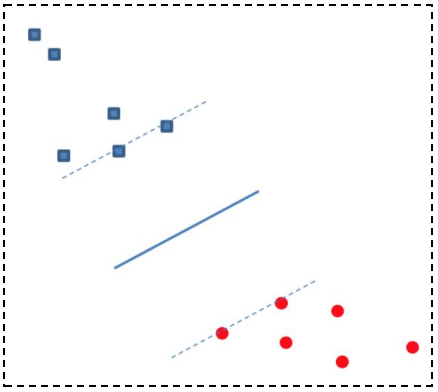
$$\frac{|w \cdot x^k + b|}{||w||} = \frac{1}{||w||} = \frac{1}{2}$$

(c) Lecture slide SVM pg.7 :

Maximizing margin is equivalent to minimizing $\frac{1}{2}w^2$

In figure.2 the margin is larger therefore $\frac{1}{2}w^2$ is smaller.

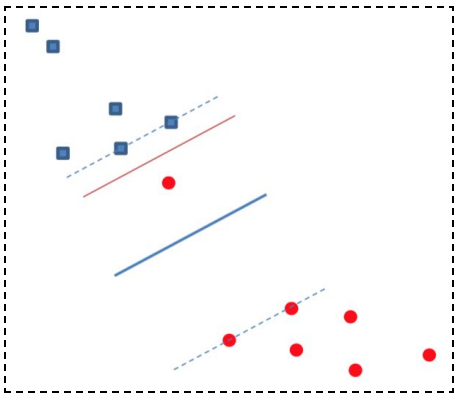
(d)



||w|| = 2

$\frac{1}{||w'||} = 4 \cdot (\frac{1}{||w||}) = 2$

(e)



SVM.pdf: More one 2nd case - margin errors.

Because x_k is on the wrong side of the separate line.

The slack variable represents the distance from variable x_k to the class's margin line, if on the wrong side of it. So in the new drawing it's value is around 1.5 (1<slack variable<2). And this also explains why the redline's cost is zero.

(f) Again, it is because the outliner point is on the wrong side of original separate line. We need to update it to reduce cost.

2. (a) Half of the classifications are wrong. 50%

(b) Using confusion matrix error rate calculation:

Error rate = FP+FN/P+N

FP = 0.8*0.5 = 0.4, FN = 0.2*0.5= 0.1

P+N = 0.8 + 0.2 = 1

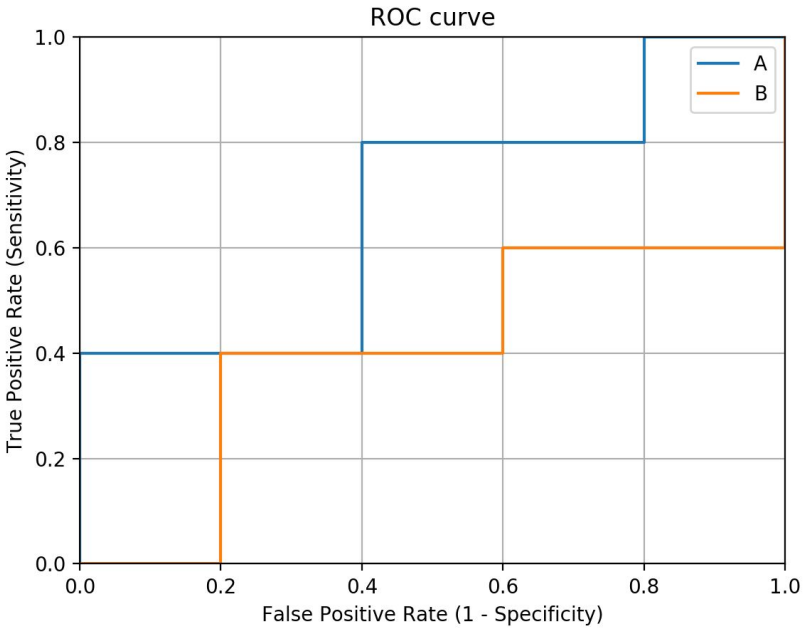
Error rate = 0.5/1 = 0.5 = <u>50%</u>

(c) FP = ⅓*1 = 1/3 FN = 0, Error rate = ⅓ /1 = ⅓ = 33.33%

(d) ⅓ * ⅔ + ⅔ * ⅓ = 4/9 = 44.44%

3. (a)

```
import pandas as pd
import sklearn.metrics
import matplotlib.pyplot as plt

A_score = [0.73,0.69,0.67,0.55,0.47,0.45,0.44,0.35,0.15,0.08]
B_score = [0.61,0.03,0.68,0.31,0.45,0.09,0.38,0.05,0.01,0.04]
y_test = [1,1,0,0,1,1,0,0,1,0]

fpr, tpr, thresholds = sklearn.metrics.roc_curve(y_test, A_score)
plt.plot(fpr,tpr,label="A")

fpr, tpr, thresholds = sklearn.metrics.roc_curve(y_test, B_score)
plt.plot(fpr, tpr,label="B")

plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.title('ROC curve')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.grid(True)
plt.legend(loc=0)
plt.show()
```



From the graph we observe that A curve is closer to the top left corner, relatively, therefore being the better classifier.

(b) precision = TP/(TP+FP) = 3/(3+1)=75%
recall = TP/P = 3/5 = 60%
Fmeasure = 2*Precision*Recall / (Precision + Recall)
= 2*0.75*0.6 / (0.75+0.6) ≈ 0.67

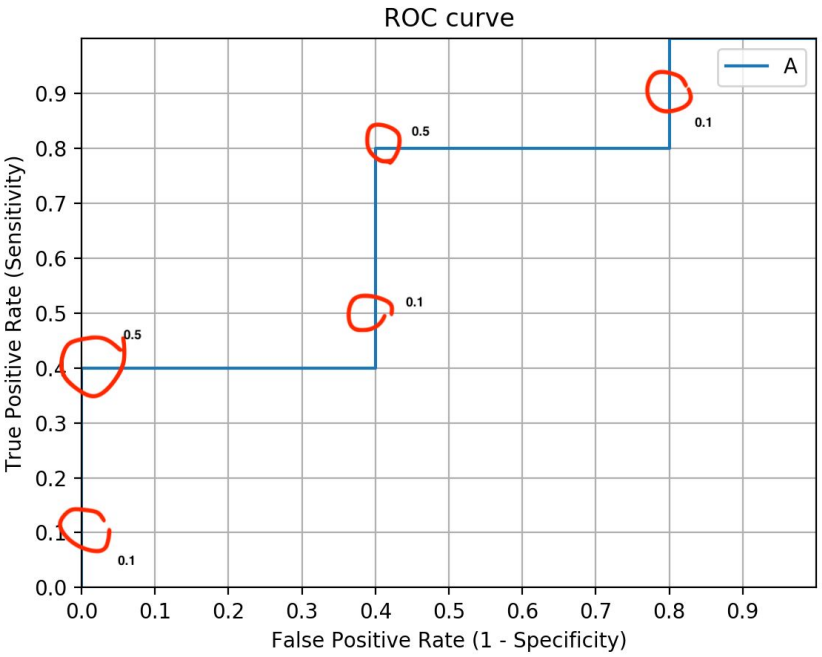
(c)

```
import pandas as pd
import sklearn.metrics
import matplotlib.pyplot as plt

A_score = [0.73,0.69,0.67,0.55,0.47,0.45,0.44,0.35,0.15,0.08]
B_score = [0.61,0.03,0.68,0.31,0.45,0.09,0.38,0.05,0.01,0.04]

true_result = [1,1,0,0,1,1,0,0,1,0]

B_result = []

for x in B_score:
    if(x>0.5):
        B_result.append(1)
    else:
        B_result.append(0)

print("B predictions are: ", B_result)
print("Confusion Matrix: ")
print(sklearn.metrics.confusion_matrix(true_result,B_result))
print("F measure: ", sklearn.metrics.f1_score(true_result,B_result))
```

```
B predictions are:  [1, 0, 1, 0, 0, 0, 0, 0, 0, 0]
Confusion Matrix:
[[4 1]
 [4 1]]
F measure:  0.28571428571428575
```

From the Fmeasures we can tell classifier A is much better with 0.67 over B's 0.29
Both results are consistent with ROC plots.

(d)



For example for a dataset for wine quality that consists of numerical values 0.0-1.0, worst to best respectively. We can use 0.5 as the cut-off value here as anything greater is determined as good, anything lower means bad. However if a dataset for contagious disease outbreaks in a certain area is presented, and the binary class is whether the health administration is going to enhance its local medical resource. In this case the real labels may be numerical values from 0.0-1.0, the threshold can be set as low as 0.1 because essentially anything greater than 0 is positive

4. a)
Let C1 be the kth frequency set and L1 be the saved frequency set after pruning. Minsup is the number of transactions the itemset appears in the data.

Iteration 1:

C1	minsup	L1	minsup
{a}	5	{a}	5
{b}	7	{b}	7
{c}	5	{c}	5
{d}	8	{d}	9
{e}	6	{e}	6

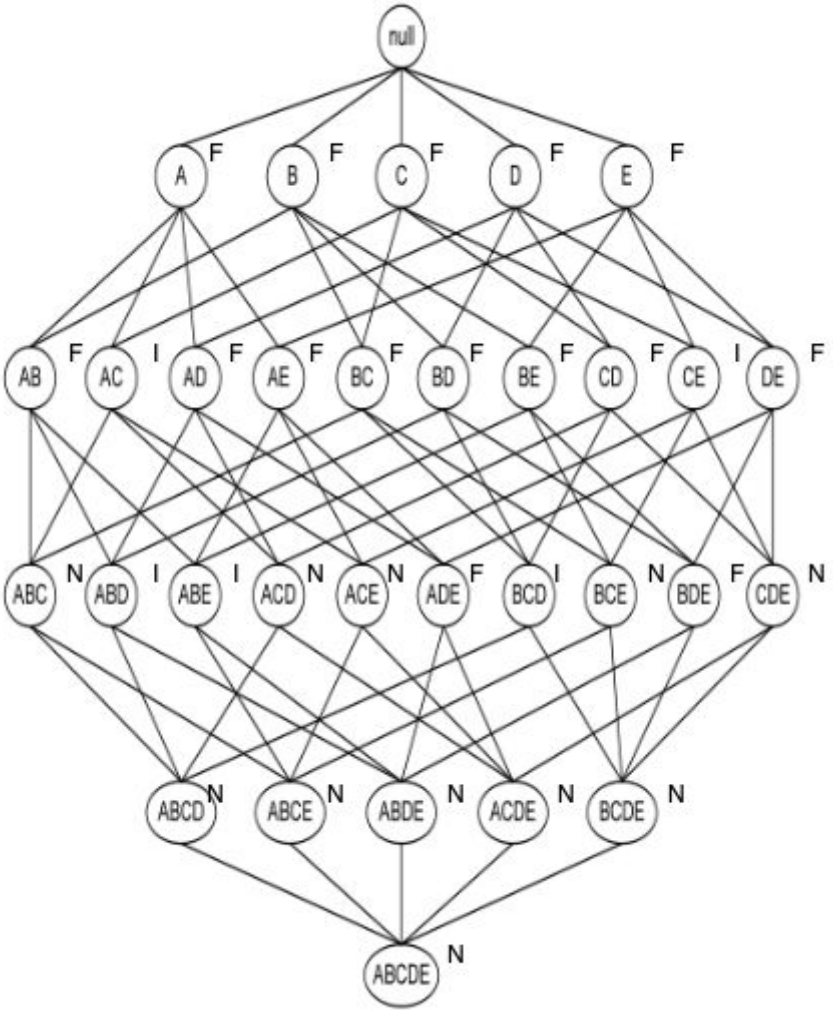
Iteration 2: Sets pruned are {a,c},{c,e}

C2	minsup	L2	minsup
{a,b}	3	{a,b}	3
{a,c}	2	{a,d}	4
{a,d}	4	{a,e}	4
{a,e}	4	{b,c}	3
{b,c}	3	{b,d}	5
{b,d}	5	{c,d}	4
{b,e}	4	{b,e}	4
{c,d}	4	{d,e}	5
{c,e}	2		
{d,e}	5		

Iteration 3: Sets pruned are {a,b,d},{a,b,e},{b,c,d}

C3	minsup	L3	minsup
{a,b,d}	2	{a,d,e}	3
{a,b,e}	2	{b,e,d}	3
{a,d,e}	3		
{b,c,d}	1		
{b,e,d}	4		

The Lattice Structure is then lab



b) 15/(2^5) = 0.46

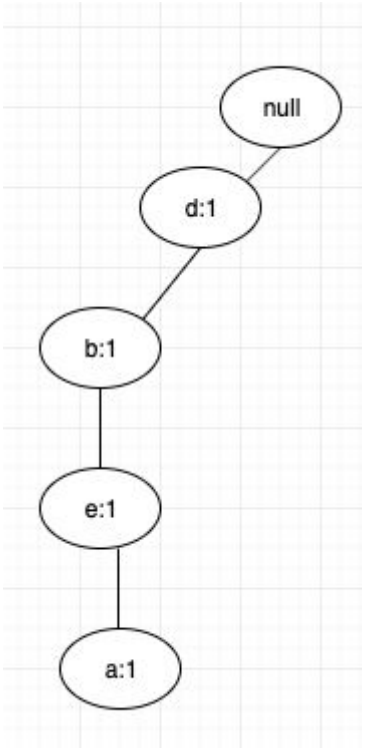
c) 1 - 0.46 = 0.54

d) 5/(2^5) = 0.156

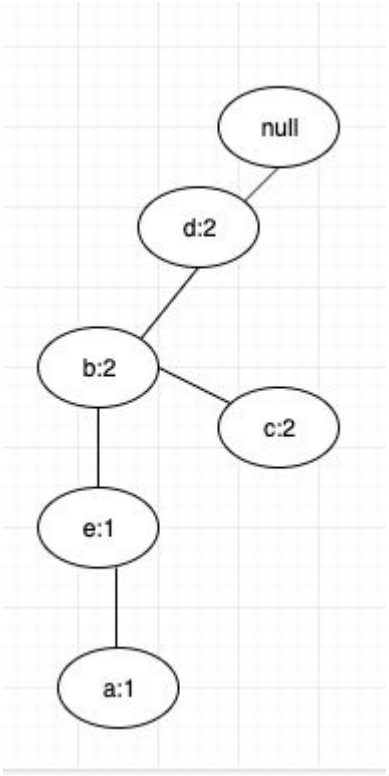
5.

Header		ReOrdered Data set
{d}	8	{d,b,e,a}
{b}	7	{d,b,c}
{e}	6	{d,b,e,a}
{a}	5	{d,e,c,a}
{c}	5	{d,b,e,c}
		{d,e,b}
		{d,c}
		{b,c,a}
		{d,e,a}
		{d,b}

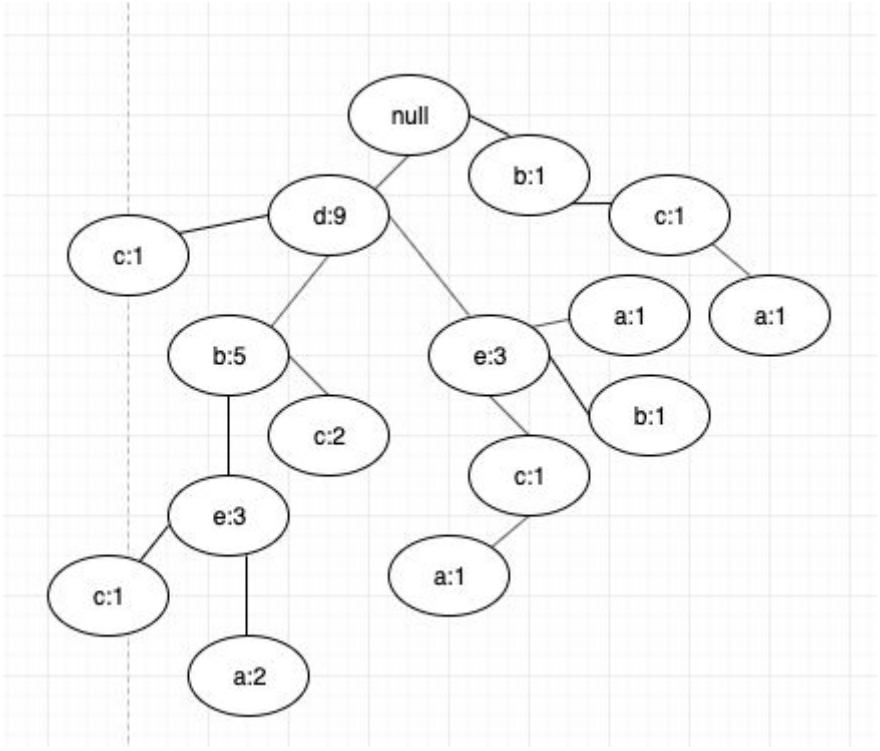
Iteration 1:



Iteration 2:



…...
Iteration 10:



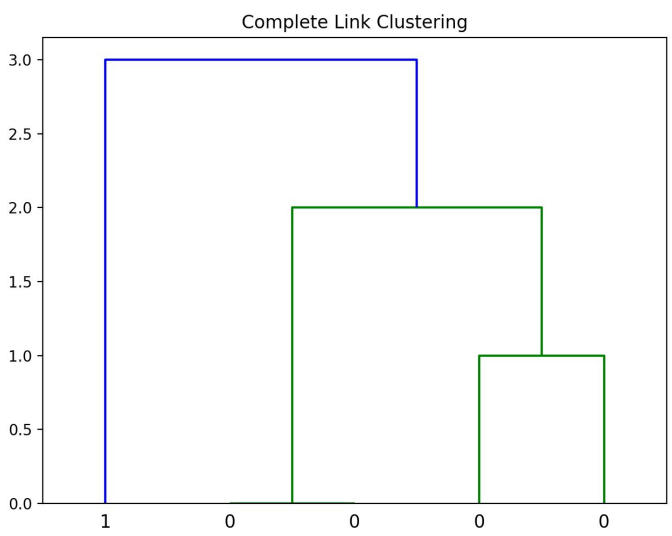
Using the FP-Tree produces the following frequent item set results

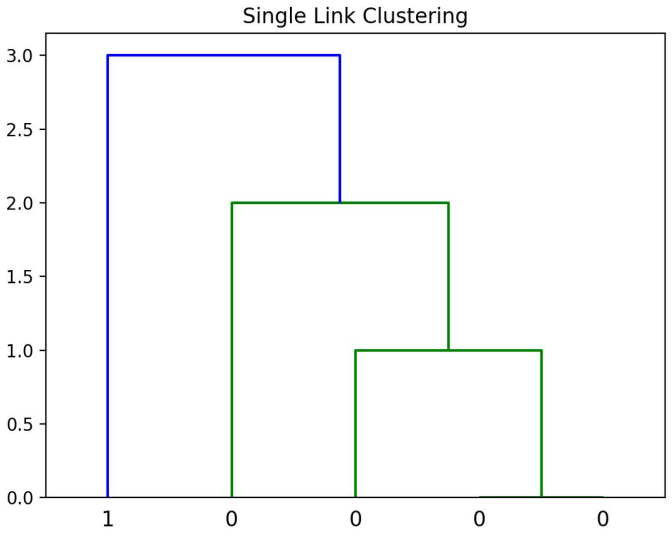
	support	itemsets
0	0.9	(d)
1	0.7	(b)
2	0.6	(e)
3	0.5	(a)
4	0.5	(c)
5	0.6	(b, d)
6	0.6	(e, d)
7	0.4	(b, e)
8	0.4	(b, e, d)
9	0.4	(e, a)
10	0.4	(a, d)
11	0.3	(b, a)
12	0.4	(e, a, d)
13	0.4	(c, d)
14	0.3	(b, c)

6.

```python
import numpy as np
from matplotlib import pyplot as plt
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram

similarity_matrix = [   [1.0,0.1,0.41,0.55,0.35],
                        [0.1,1.0,0.64,0.47,0.98],
                        [0.41,0.64,1.00,0.44,0.85],
                        [0.55,0.47,0.44,1.00,0.76],
                        [0.35,0.98,0.85,0.76,1.00]
                    ]

###     Function to plot dendrogram from sklearn.cluster agglomerativeclustering model output
###
###     https://github.com/scikit-learn/scikit-learn/blob/70cf4a676caa2d2dad2e3f6e447
###     8d64bcb0506f7/examples/cluster/plot_hierarchical_clustering_dendrogram.py
def plot_dendrogram(model, **kwargs):

    # Children of hierarchical clustering
    children = model.children_

    # Distances between each pair of children
    # Since we don't have this information, we can use a uniform one for plotting
    distance = np.arange(children.shape[0])

    # The number of observations contained in each cluster level
    no_of_observations = np.arange(2, children.shape[0]+2)

    # Create linkage matrix and then plot the dendrogram
    linkage_matrix = np.column_stack([children, distance, no_of_observations]).astype(float)

    # Plot the corresponding dendrogram
    dendrogram(linkage_matrix, **kwargs)


model = AgglomerativeClustering(affinity='precomputed',linkage='single/complete').fit(similarity_matrix)
plt.title("Single/Complete Link Clustering")
plot_dendrogram(model, labels=model.labels_)
plt.show()
```

Dendrogram outputs:





7. **(a)**

1. First we we found all the movies that each person has in common with M. Phillips.
2. Next we calculated the mean scores for how each person rated those movies, and then used that we calculated their centralized scores.
3. Finally, we would calculate their pearson “r” score using the similarity formula

$$sim_{x,y} = \frac{\sum_{i=1}^{m} (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{m} (x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{m} (y_i - \bar{y})^2}}$$

4. Finally we calculate the total scores of the people that have a positive “r” score, take their score for the film and multiply it by their similarity, sum those values, and then divide it by the total similarity sum to get a final prediction. As we can see the final prediction value in this case, the final prediction score is ~2.72. Please see the figures below for the complete calculations.

	M. Philips	C. Puig	M. Philips Centralized	C. Puig Centralized		
Snakes on a Plane	3	3.5	-0.5	-0.5		
Superman Returns	3.5	4	0	0		
The Night Listener	4	4.5	0.5	0.5		
Mean	3.5	4			1	Pearson sim
	M. Philips	L. Rose	M. Philips Centralized	C. Puig Centralized		
Lady in the Water	2.5	2.5	-0.75	-0.625		
Snakes on a Plane	3	3.5	-0.25	0.375		
Superman Returns	3.5	3.5	0.25	0.375		
The Night Listener	4	3	0.75	-0.125		
Mean	3.25	3.125			0.40452	Pearson sim
	M. Philips	Toby	M. Philips Centralized	Toby Centralized		
Snakes on a Plane	3	4.5	-0.25	0.25		
Superman Returns	3.5	4	0.25	-0.25		
	3.25	4.25			-1	Pearson sim
	M. Philips	Mick LaSalle	M. Philips Centralized	Mick LaSalle Centralized		
Lady in the Water	2.5	3	-0.875	0		
Snakes on a Plane	3	4	-0.375	1		
Superman Returns	4	2	0.625	-1		
The Night Listener	4	3	0.625	0		
	3.375	3			-0.544331	Pearson sim

	M. Philips	Gene Seymour	M. Philips Centralized	Gene Seymour Centralized		
Lady in the Water	2.5	3	-0.875	0.25		
Snakes on a Plane	3	3.5	-0.375	0.75		
Superman Returns	4	1.5	0.625	-1.25		
The Night Listener	4	3	0.625	0.25		
	3.375	2.75			-0.57735	Pearson sim
	M. Philips	Jack Matthews	M. Philips Centralized	Jack Matthews Centralized		
Lady in the Water	2.5	3	-0.875	-0.75		
Snakes on a Plane	3	4	-0.375	0.25		
Superman Returns	4	5	0.625	1.25		
The Night Listener	4	3	0.625	-0.75		
	3.375	3.75			0.406181	Pearson sim
Person	Similarity	Score	Weighted Score			
C. Puig	1	2.5	2.5			
L. Rose	0.40452	2.5	1.0113			
Jack M.	0.406181	3.5	1.4216335			
Total Score	4.9329335					
Similarity Sum	1.810701					
Total/Sim.Sum	2.724322514					

b)

1. To calculate the bias for Lisa Rose for the first iteration, we followed the provided formula: br = br + gamma*((mr-(mu+br))- lambda*br), where br is initialized to 0, gamma is initialized to 0.1, lambda is initialized to 0.1 and mr = movie rating.
2. We then repeated this process for each movie Lisa Rose had watched, to get the Bias for Lisa Rose at the end of the first pass of the data. (Please see the figure below for the full calculations.

(b)

```
gamma=0.1
lambda=0.1

mu = 3.23

br = 0 %and all bi's are 0 initially. The predicted r_ui is mu+bu+bi or just mu+bu since bi's are 0 initially.

br = br + gamma*((2.5-(mu+br))- lambda*br)
         -0.072857
br = br + gamma*((3.5-(mu+br))- lambda*br)
           -0.0377

br = br + gamma*((3.0-(mu+br))- lambda*br)
br =-0.0377 + 0.1*((3-(3.23-0.0377))- 0.1*-0.0377)
         -0.056553

br = br + gamma*((3.5-(mu+br))- lambda*br)
br = -.056553 + .1*((3.5-(3.23+-.056553))- .1*-.056553)
       -0.02333217

br = br + gamma*((2.5-(mu+br))- lambda*br)
br = -0.02333217 + 0.1*((2.5-(3.23+-.02333217))- 0.1*-0.02333217)
      -0.093765631

br = br + gamma*((3.0-(mu+br))- lambda*br)
br = -0.093765631 + 0.1*((3.0-(3.23+-0.093765631))- 0.1*-0.093765631)
      -0.106451412
```