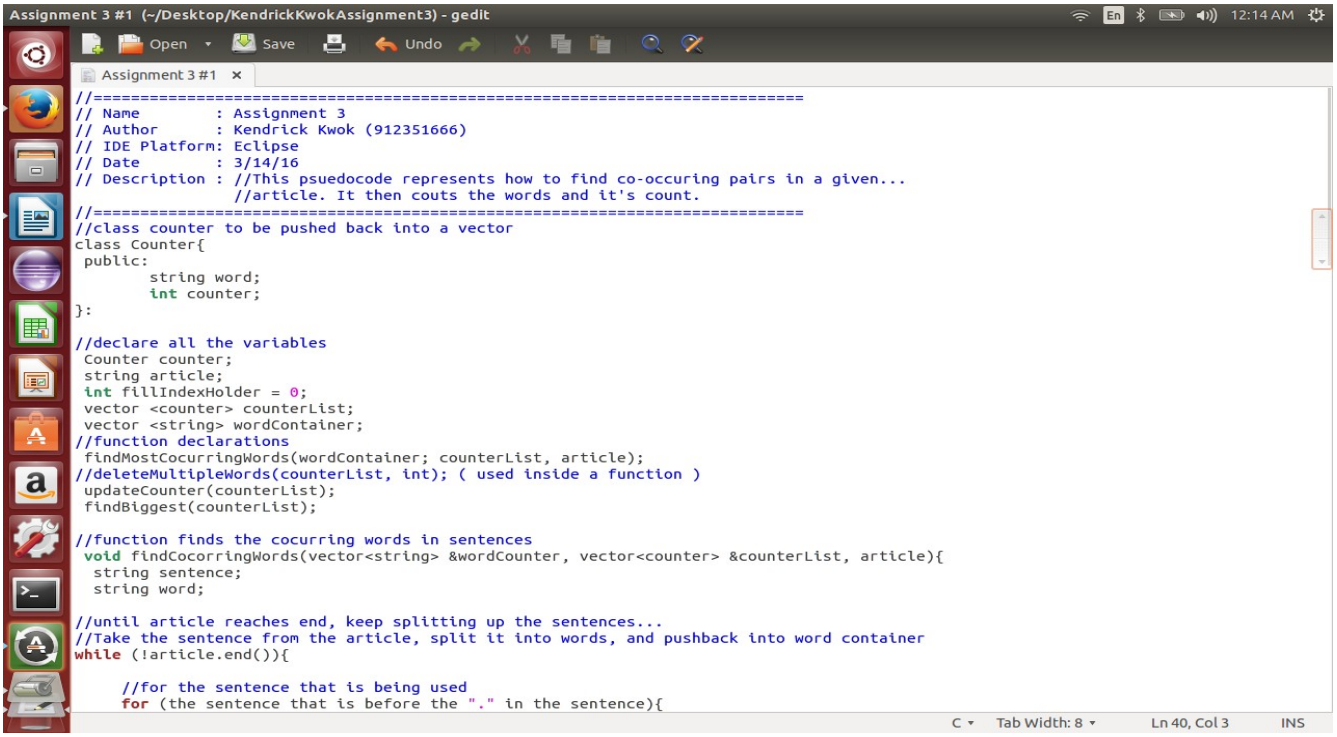
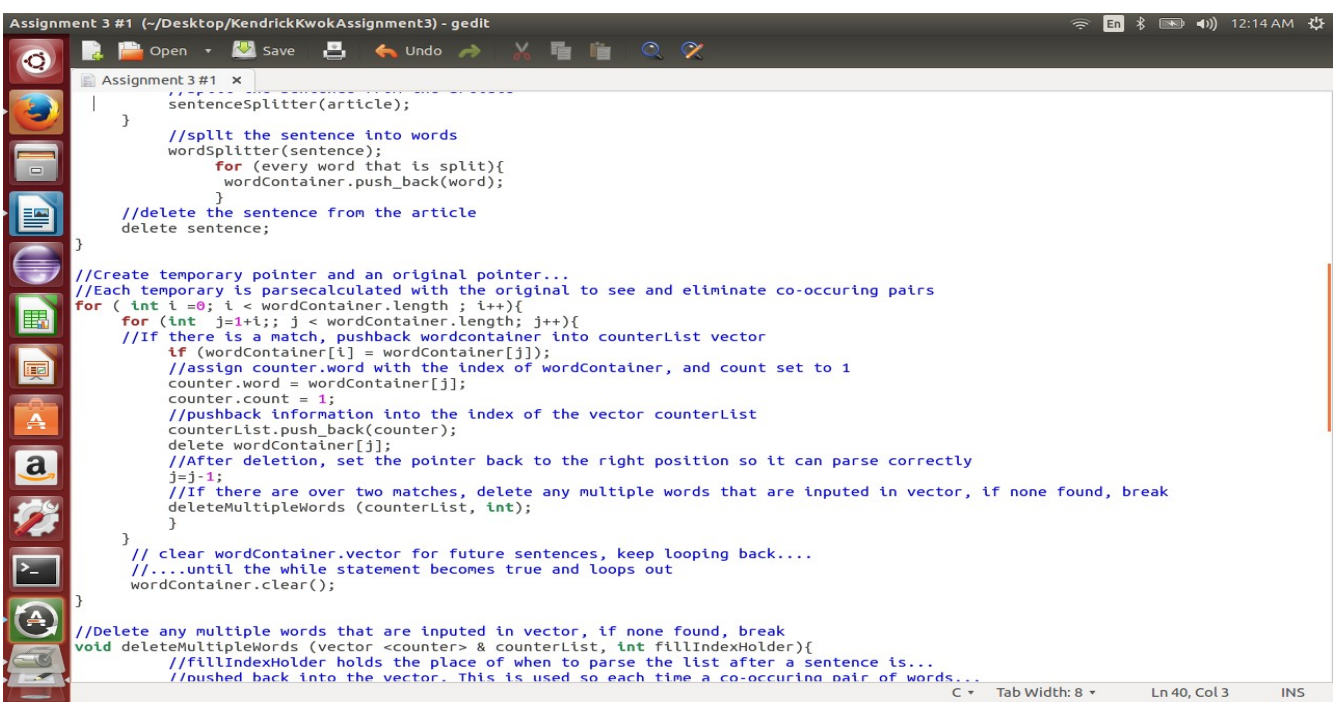
//==========
// Name : Assignment 3
// Author : Kendrick Kwok (912351666)
// Platform: :Linux text file, used GEDIT to open these text files
// Date : 3/14/16
// Description : //This assignment was to write pseudocode for the different prompts.
//==========

#1: //This psuedocode represents how to find co-occuring pairs in a given article. It then couts the words and it's count. I wrote the pseudocode using a LINUX text file, using GEDIT to open the files. Here should how the file should look when it is opened.





Assignment 3 #1 (~/Desktop/KendrickKwokAssignment3) - gedit

```
        //found from the previous sentence
        for ( int i = fillIndexHolder; i < counterList.length ; i++){
            for (int  j=1+i;; j < counterList.length; j++){
                if (there is more than one index with the same words duplicated in counterList vector){
                counterList(fillIndexHolder+i) = counterList(j)
                delete counterList(j);
                j=j-1;
                fillIndexHolder = fillIndexHolder + counterList.length();
                }
                else{
                 break;
                }
            }
        }
    }
}

//update the counter so each similar words would be added to count after all the co-occuring words...
//... are added into the vector
void updateCounter(vector <counter>&counterList){
for ( int i =0; i < counterList.length ; i++){
    for (int  j=1+i;; j < counterList.length; j++){
    //parse through the list, if common word is found in vector, update counter +1
        if (counterList[i].word = counterList[j].word){
        counterList[i].counter = counter + 1;
        //erase the word and count from the list
        counterList.erase (counterList.begin(),counterListr.begin()+j)
            //j is minused one so it can keep parsing after deletion
            j=j-1;
        }
    }
}

//Find the biggest amount of count inside the vector counterList
void findBiggest(vector <counter>&counterList){
    //Original is put in the very first index
    original = counterList[0].count;
    for (unsigned int originalIndex=1; originalIndex < counterList.size(); originalIndex++){
```

C ▾ Tab Width: 8 ▾ Ln 40, Col 3 INS

Assignment 3 #1 (~/Desktop/KendrickKwokAssignment3) - gedit

```
//update the counter so each similar words would be added to count after all the co-occuring words...
//... are added into the vector
void updateCounter(vector <counter>&counterList){
for ( int i =0; i < counterList.length ; i++){
    for (int  j=1+i;; j < counterList.length; j++){
    //parse through the list, if common word is found in vector, update counter +1
        if (counterList[i].word = counterList[j].word){
        counterList[i].counter = counter + 1;
        //erase the word and count from the list
        counterList.erase (counterList.begin(),counterListr.begin()+j)
            //j is minused one so it can keep parsing after deletion
            j=j-1;
        }
    }
}

//Find the biggest amount of count inside the vector counterList
void findBiggest(vector <counter>&counterList){
    //Original is put in the very first index
    original = counterList[0].count;
    for (unsigned int originalIndex=1; originalIndex < counterList.size(); originalIndex++){
    //if the counterList count starting from 0 is bigger the ones after, assign original to be...
    //biggest count
         if (counterList[originalIndex].count > original){
          original = list[originalIndex].count;
         }
    }
    //Loop through the whole vector, and if original (original = largest) is the same amount...
    //of counts as the other elements inside the vectors, cout the vector element with same value
    for (int i = 0; i < counterList.size(); i++){
        if (original == counterList[i].count){
         cout << "Your found the most frequent co-occuring word is: " counterList[i].word << endl;
         cout << "Your count is: " << original << endl:
        }
    }
}
```

C ▾ Tab Width: 8 ▾ Ln 40, Col 3 INS

#2. This psuedocode represents how to find products that is considered frequent (i.e products that are transactioned) more then 5% of the total amount of transactions. I used linux txtfile, and gedit to open it.

Assignment 3 #2 (~/Desktop/KendrickKwokAssignment3) - gedit

```
//=======//============================================================================
// Name        : Assignment 3
// Author      : Kendrick Kwok (912351666)
// IDE Platform: Eclipse
// Date        : 3/14/16
// Description : //This psuedocode represents how to find products that is considered...
                 //frequent, i.e products that are transactioned....
                 //more then 5% of the total amount of transactions
//============================================================================


class productPairs{
     string products;
     int transactions;
}

//Declare variables used in the main function
     int totalTransaction;
     int calculatedFivePercent;
     min_freq = .05;

//put the class into a vector
     vector <productPairs> productVector;

//Declare an object for productPairs
     productPairs productsBought;

//Assume that you have a list of products with the number of transactions already tied to it called "productPairs"
//Put every single product with the transactions into vector called productVector
for(int i =0; i < productVector(); i++){
     productsBought.products.push_back(productVector);
     productsBought.transactions(productVector);
}

//for every product inside the productVector, take each transaction and add it
for(int i = 0 ; i < productVector.length(); i++){
     totalTransaction = totalTransaction + productVector(i).transactions;
}
```

Assignment 3 #2 (~/Desktop/KendrickKwokAssignment3) - gedit

```
     min_freq = .05;

//put the class into a vector
     vector <productPairs> productVector;

//Declare an object for productPairs
     productPairs productsBought;

//Assume that you have a list of products with the number of transactions already tied to it called "productPairs"
//Put every single product with the transactions into vector called productVector
for(int i =0; i < productVector(); i++){
     productsBought.products.push_back(productVector);
     productsBought.transactions(productVector);
}

//for every product inside the productVector, take each transaction and add it
for(int i = 0 ; i < productVector.length(); i++){
     totalTransaction = totalTransaction + productVector(i).transactions;
}

//find what is 5% of the totalTransactions made;
     calculatedFivePercent = totalTransaction * min_freq;

//For every index of productVector, see if it is actually 5%. If it is under 5%, delete from the vector list
for (int i =0; i< productVector.length(); i++){
     //If the products total transaction is less then the calculated five percent, delete from vector
     if (productVector(i).totalTransaction < calculatedFivePercent){
      productVector.erase(productVector.begin + i;
      i=i-1;
     }
}

//Parse through the vector productVector and print out all the products that is more frequent
for (int i =0; i< productVector.length(); i++){
 cout << productVector(i).product << " is considered more frequent!" << endl;
}
```