CSE5243 **Student Name:** _____
Autumn 2019 **Student ID:** _____
Programming Assignment 2
Due date: Thursday, Nov 7, 12:44pm

Instructions:

In this programming assignment, you will implement the Apriori algorithm. The total point is 146 and you may get some points as bonus. Good luck!

Note: This is not an easy programming assignment. Start earlier!

Grade Table (for TA/grader use only)

Question	Points	Score
1	11	
2	22	
3	10	
4	103	
5	0	
Total:	146	

In this programming assignment, you will implement the Apriori algorithm to find frequent itemsets from a transaction database.

1. (11 points) Your executable should have the name "Apriori", and it takes arguments as follows:

   ./Apriori -database_file=database.txt -minsupp=alpha -output_file=output.txt

   where database.txt (2 points) is the input file with a database of transactions (its format will be discussed later), alpha (2 points) is a value in [0, 1] and is the minimum support (in percentage) that a frequent itemset needs to have, and output.txt (2 points) is the output file into which your algorithm outputs the frequent itemsets (its format will be discussed later).

   You need to provide a README file in which you provide a command line (5 points) similar to the above one such that by copy-pasting your commend line on stdlinux, the TA should be able to run your algorithm. That also means, you have to make sure your algorithm runs on stdlinux.

2. (22 points) You need to implement a function named **read_database** (20 points) to read in the transactions from the database file. You are free to use whatever data structures as you see fit to represent the transactions, items and itemsets. In your README file, you need to specify what data structures you use for the representations (2 points).

   Transaction database file format: an example of the transaction database file is as follows:

   6 5
   0 2 3
   0 1 2 3
   0 3 4
   2 4
   1 2 4
   2 3 4

   that is, for a transaction database of n transactions ($n = 6$ in the above example) over m items ($m = 5$ in the above example), there will be $n + 1$ lines in the database file (7 lines in the above example). The first line has exactly 2 numbers, the first number is the number of transactions in the database (6 in the above example), and the second number is the number of items in the database (5 in the above example). In the rest n lines, each line represents a transaction. Each transaction has a number of items, for example, the first transaction in the above example has item 0, 2 and 3 (C-style indexing)

3. (10 points) You need to have a function named **apriori** (10 points) according to the following Apriori algorithm pseudo code

```
def apriori(database, minsupp, output_file):

k=1

# generate F1 = {frequent 1-itemsets}
generate_F1()

Repeat until Fk is empty

    # Generate Lk+1 from Fk
    generate_candidate()

    # Prune candidate itemsets in Lk+1 containing
    # subsets of length k that are infrequent
    prune_candidate()

    # Count the support of each candidate
    # in Lk+1 by scanning the DB
    count_support()

    # Eliminate candidates in Lk+1 that are
    # infrequent, leaving only those that are frequent => Fk+1
    eliminate_candidate()

    k ++

#output F1, F2, ..., Fk to output_file
output_freq_itemsets()

return
```

4. (103 points) That is, you need to implement functions **generate_F1()** (10 points), **generate_candidate()** (30 points), **prune_candidate()** (15 points), **count_support()** (30 points) and **output_freq_itemsets()** (10 points). For these functions, you are free to use whatever arguments as you see fit and you need to specify the data structures you use in the first 4 functions in the README file (8 points). Note that you need to use all the given function names so that the TA can quickly identify your functions.

   Output file format: the output file should have the same format as the input transaction database file.

5. * Bonus credits: You will get bonus credits proportionally to the run-time performance of your implementation compared to the other students. If your run-time performance ranks at x-th in the class, you will get $(1 - \frac{x}{y}) \times 15$ bonus credits, where y is the total number of students whose algorithms are correct. Note that if your algorithm is not correct, you will not get any bonus credits for run-time performance.

   What to submit:

   1. Your source code in a zipped file. You should name your zipped file as X_Y_PA2.tar.gz

where X is your first name (capital letters) and Y is your last name (capital letters). Please following the naming schemes specified, otherwise, **20** points will be taken off. This is for the TA to quickly find your submission.

2. A README file with required content as described earlier.
3. It is mandatory that at least 1/4 of the lines (not including empty lines) in your code should be dedicated to comments. Otherwise, **20%** of the points that you will get will be taken off (non-genotiable!)

Grading: you need to make sure your code runs on stdlinux. If your code does not run on stdlinux, you will get points based on your implementation of the required functions. However, you will lose 20% of the points that you can get from the implementation.