O COMP 215 Algorithms Θ
Lab 5

This work should, if possible, be completed during the lab and submitted electronically before 11:59pm Thursday. I kept this one short since you have an assignment to hand in on Friday.

I noticed during the last lab that many of you had trouble with file output, and especially file input. So this time, we are going to do file input and output again, but this time with an additional twist: the input and output from and to the file will be required to have a certain format.
Other than that, you will probably notice that not much else makes a lot of sense here. Clearly, this whole exercise was designed by a complete idiot. You are not required to create any special data structures to hold the data contained in the file, as long as your program has the proper input and output behaviour.

1. First download the file"InputData.zip" from onCourse. Extract the file "barn.moo" from this .zip file. This will be your initial input file.
   The first line from the file barn.moo contains an integer. This integer is the remaining number of lines in the file. In the remainder of this text, I will be referring to this number as `nblines`. Do not hardcode that integer in your program, I will be testing your program with files of variable length.
   The following `nblines` of the file have the following format:
   woof [integer] meow meow [integer] quack oink quack [integer] roar

2. Write a function that reads all the integers from this file (we do not care about all the animal noises) and write them all in a new file "reverse.dat". As its name indicates, the order of the lines should be reversed in this file, but the order of the integers on each line should be the same as in the original file. The first line of the file "reverse.dat" should be:
   There are [nblines] lines in this file.
   and the following `nblines` of the file should be formatted exactly as follows (with no white space):
   [integer],[integer],[integer]

3. Write a function that reads the file "reverse.dat" and writes in a new file "swap.dat". The first line of this new file should be
   There are also [nblines] lines in this file.
   and the remaining lines of the file should be formatted as follows:
   First integer: [integer] Second integer: [integer] Third integer: [integer]
   with the additional twist that the integers in the first line of "swap.dat" should be from the second line of "reverse.dat", the integers in the second line of "swap.dat" should be from the first line of "reverse.dat", the integers from the third line of "swap.dat" should be from the fourth line of "reverse.dat", the integers from the fourth line of "swap.dat" should be from the third line of "reverse.dat", the integers from the fifth line of "swap.dat" should be from the sixth line of "reverse.dat", the integers from the sixth line of "swap.dat" should be from the fifth line of "reverse.dat" and so on.
   If "reverse.dat" contained an even number of lines, that will be all of it. If it contained an

odd number of lines, then the second to last line of "swap.dat" should be formatted like all the others, except that its first integer should be the number of even integers in all of "reverse.dat" (not counting `nblines`), the second integer should be the number of integers in "reverse.dat" that are divisible by 3, and the third integer should be the number of integers in "reverse.dat" that are divisible by 5; the integers from the last line of "swap.dat" should be from the last line of "reverse.dat".