# Business Information Systems

The following questions should be completed and returned to the HR office within the next 24 to 48 hours. If an external problem arises that prevents you from completing the questions in the allotted time, please make us aware so that necessary arrangements can be made. If a question does not explicitly state a development language to use, then you may use whichever language you would like. If you are unsure of any instructions in this document, please state all of your assumptions so that we can grade your response on that understanding.

Read each question completely  
Check to make sure you have answered all three questions

# Fibonacci

### Problem

Write a program that will produce the first n Fibonacci numbers.

Fibonacci numbers form a sequence such that each number is the sum of the two preceding numbers, starting with 0 and 1. There are a few different ways to implement a function that generates the Fibonacci sequence including, but not limited to, recursion and storing results. Feel free to choose whichever method you'd like, just make sure you have a reason for doing so.

### Example

Function call: Fibonacci(10)  
Function output: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34

# 

# Word Count

Write a program to count the frequencies of unique words from standard input, then print them out with their frequencies, ordered from most frequent to least frequent.

For example, given this input:

The foo the foo the  
defenestration the

The program should print the following:

the 4  
foo 2  
defenestration 1

### Input File

<https://www.gutenberg.org/files/1/1-0.txt>

### Constraints

* The program must normalize words to lowercase, so "The the THE" should appear as "the 3" in the output
* A word is anything separated by whitespace - ignore punctuation
* It's okay to only support ASCII for the whitespace handling and lowercase operation
* If the frequency of two words is the same, their order in the output doesn't matter
* The program should run in a single thread on a single machine
* Don't read the whole file into memory
* Use only the language's standard library functions

# 

# Balanced Parentheses

Using only standard library functionality, write a function that solves the following problem:

Given a string containing ( and ), are the parentheses in the string balanced? For the parentheses to be balanced, each open parenthesis must have a corresponding close parenthesis, in the correct order. For example:

* (a(b(c))ffff) is balanced
* (hello(world)((foo)(bar)))baz is balanced
* )bubbles(frog is not balanced
* (((strawberry) is not balanced
* t(s)b)c) is not balanced

Time and space complexity are unimportant. The function should take a single string input argument and should return a boolean value indicating whether or not the parentheses are balanced. The following test cases can be used to test the function:

* r(((Vwhh(c)A9)yY0P))
* Mp6Js((()))StuqRxRy8
* (x(N(b(3(j)E)2)Z)J)5
* ))))))))))))))))))))
* mwc(((agsnXSd)))o)Me
* gK((((Yu1)fK)p8)Fh)Y
* x(F[D{Li}S]B)R5tiTlT
* (((((((((())))))))))
* q((2t((8fm)C0)0)E)AE
* ilZUhzFKCq1AjMxqTJi3

How would you change your solution if instead of just parentheses there were also curly braces {} and square brackets []? Like before, all must have open and close pairs, and they must match up in exactly the correct order in the input string. For example:

* ([mars]{venus}) is balanced
* (a{b[c]d}e)fghi is balanced
* 1(2[3)4]5 is not balanced
* (} is not balanced
* (z)}y is not balanced
* ({}aaa is not balanced

Here are some test cases:

* ({[({[[({[]})]]})]})
* oXK(Xl[1m{}2]HJSZNm)
* (((([[fVmz2qmz}WJfVp
* UjuTO2z2qtBn82qpFYS9
* 2R(Tf{{{Gb]]FlwX)m7g
* (G{M[L{z(w)N}d]x}m)l
* MWfv2JZFVHINWO8uTcyd
* yhj{{{{{}}}}}Wz[]x()
* [[[[[[[[[[[[[[[[[[[[
* l0SOvapJmIrjel4qMh96
* (a(b(c))[)df[]]f)