CMPSC460 Homework 3 Spring 2020

*** Submit your Haskell program (.hs) on Canvas.

Problem description: A directed graph can be implemented as a list, whose elements are pairs of nodes. Assume the nodes are integers. In Haskell, you can define a graph as follows:

```
type Node = Integer
type Edge = (Integer, Integer)
type Graph = [Edge]
type Path = [Node]
g :: Graph
g = [ (1, 2), (1, 3), (2, 3), (2, 4), (3, 4) ]
h :: Graph
h = [ (1, 2), (1, 3), (2, 1), (3, 2), (4, 4) ]
```

1. Write a function named nodes to get the list of nodes (increasing) of a graph. For example, nodes g => [1, 2, 3, 4] .

2. Write a function named adjacent that takes a node N and a graph G and returns a list of all nodes that are connected to N such that N is the source. For example, adjacent 2 g => [3, 4], adjacent 4 g => [] , adjacent 4 h => [4].

3. Write a function named detach that takes a node N and a graph G and returns the graph formed by removing N from G. For example, detach 3 g => [(1, 2), (2, 4)].

4. Write a function named paths that accepts two nodes V1 and V2 (and a graph G) and returns a list of all possible cycle-free paths from V1 to V2. For example, paths 2 2 g => [[2]], paths 3 2 g => [], paths 1 4 g => [[1, 2, 3, 4], [1, 2, 4], [1, 3, 4]].