#lang racket

(define (make-graph graph) ;The define function specifies the head-argument make-graph and graph.

(cond [(empty? graph)] ;The cond function tests to see if the graph is empty.

[(cons? graph) ((first (first graph)); This provides the first node in graph list.

(second (first graph)) ;This provides a list of adjacent nodes

(make-graph (rest graph)))])) ;This lists the rest of the nodes in graph

(define (neighbors a-node a-graph)

(cond [(empty? a-graph) (error "Node not in graph")] ;This loads gives an error when there is no node in the graph.

[(symbol=? a-node (first (first a-graph))) (second (first a-graph))] ;This checks to see if node is a neighbor of first node and second node. If not, it reports a false.

[else (neighbors a-node (rest a-graph))])) ;This checks for the of the remaining nodes in the graph.

(define (find-route origination destination graph)

(cond [(symbol=? origination destination) (list origination)] ;This checks the list of objects in the origination to destination are true or false.

[else (local [(define nbrs (neighbors origination graph)) ;Define what neighbors can be defined on the graph.

(define route (find-route/list nbrs destination graph))] ;This determines the route that lists from the beginning of the graph

(cond [(false? route) route] ;This determine if the route of the origination to destination exists.

[else (cons origination route)]))])) ;If the route it reports the last remaining nodes.

(define (find-route/list los destination graph)

(cond [(empty? los) false] ;This determines if the los is empty. If so, reports a false.

[else (local [(define route (find-route (first los) destination graph))] ;This first set of routes from the destination nodes.

(cond [(false? route) (find-route/list (rest los) destination graph)] ;This defines the list of second routes from the destination.

[else route]))])) ;This list any other routes found from the destination.