```
$nlist=Hash.new{|h,k| h[k]=Array.new}   #hash table used to store list of roads that
connect a node with another node
$interHash=Hash.new{|roads,node| roads[node]=Array.new} #list of intersections to
allow for simple lookup

def find        #allows user to see if a given road is in the database
        puts "Enter name you would like to search for without quotation marks."
        term = gets.chomp
        puts "\n\nSearch results:"
        for i in 0..$streets.length-1
                if $streets[i].include? term
                        puts $streets[i]
                end
        end
end

def find_intersection road1, road2      #given two roads road1 and road2, the number
of the node where they intersect is returned
        temp=Array.new
        for i in 0..$interArray.length-1
                if $interArray[i][1].include? road1 and $interArray[i][1].include?
road2
                        temp << $interArray[i][0]
                end
        end
        if temp.length > 1
                puts "There are multiple intersections where these two roads meet.
Please specify a third street."
                third = gets.chomp
                temp=[]
                for i in 0..$interArray.length-1
                        if $interArray[i][1].include? road1 and
$interArray[i][1].include? road2
                                temp << $interArray[i][0]
                        end
                end
        end
        return temp[0]
end

def find_comrade(node, path)    #returns a list of the nodes that are connected to
input node
        temp = Array.new
        for i in 0..$nlist[node].length-1
                temp << $nlist[node][i][1][1]
        end
        return temp.delete_if{|w| path.include? w}
end

def goRom from, to, limit       #given origin from, destination to, and how deep the
search can go, returns the true if a path is found, false if it is not
        depth=0
        while depth<limit

        $current = $path.shift
                if $current[-1]==to
```

```
                                return true
                        else
                                temp = find_comrade $current[-1], $current
                                if(temp.length==0)
                                        return false
                                end
                                for n in 0..temp.length-1
                                        $path<< (Array.new($current) << temp[n])
                                end
                        depth+=1
                        end
        end
        return false
end

$streets=Array.new

#info in nodes:
#key=node number
#[[lat,long],[road name/type, number of node connected, lat, long]]
nodes=Array.new

#following block organizes all the data from the road database
cities=File.read("roadList.csv").chomp.split("\n")
count=0
cities.each{|array|
                a,b,c,d,e,f,g=array.split "\",\""
        a.delete! "\""          #name/type
        b.delete! "\""          #first lat
        c.delete! "\""          #first long
        d.delete! "\""          #second lat
        e.delete! "\""          #second long
        f.delete! "\""          #first node
        g.delete! "\""          #second node

        b=b.to_f
        c=c.to_f
        d=d.to_f
        e=e.to_f
        f=f.to_i
        g=g.to_i

        $streets << a   #creates a list of streets to that the method find will work
        nodes << f      #creates a list of node numbers for reference
        nodes << g

        array=[[b,c],[a,g,d,e]]
        array2=[[d,e],[a,f,b,c]]



        $nlist[f] << array
        $nlist[g] << array2

}

nodes.uniq!.sort! #gets rid of all duplicates, organizes list
```

```
$streets.sort!

$interArray=Array.new

for i in 0..1000000     #organizes intersection data, creates a hash table and array
so that other methods will work
        temp=Array.new
        temp2=Array.new
        temp << i
        for t in 0..$nlist[i].length-1
                temp2 << $nlist[i][t][1][0]
        end
        if not temp2==[]
                temp << temp2
                $interHash[i] << temp2
                $interArray << temp
        end
end

#Following portion gathers information from the user

puts "You will be asked for two pairs of roads that intersect.  The shortest route
between these two intersections will be printed out."

first="\look"
second="\look"

puts "First intersection."

while true
        puts "First road.  Please enter as \"Road Name Road Type\", without quotation
marks.  Ex: Braddock Rd, Edsall Way.\nIf you would like to look up a road, enter
\"\look\", without quotation marks."

        first=gets.chomp
        if first=="\look"
                find
        end
        if $streets.include?(first)
                break
        end
end

while true
        puts "Second road.  Please enter as \"Road Name Road Type\", without the
quotation marks.  Ex: Braddock Rd, Edsall Way\nIf you would like to look up a road,
enter \"\look\", without quotation marks."

        second=gets.chomp
        if second=="\look"
                find
        end
        if $streets.include?(second)
                break
        end
end
```

```ruby
nodeFirst = find_intersection first, second

first="\look"
second="\look"

puts "Second intersection."

while true
        puts "First road.  Please enter as \"Road Name Road Type\", without quotation
marks.  Ex: Braddock Rd, Edsall Way.\nIf you would like to look up a road, enter
\"\look\", without quotation marks."

        first=gets.chomp
        if first=="\look"
                find
        end
        if $streets.include?(first)
                break
        end
end

while true
        puts "Second road.  Please enter as \"Road Name Road Type\", without the
quotation marks.  Ex: Braddock Rd, Edsall Way\nIf you would like to look up a road,
enter \"\look\", without quotation marks."

        second=gets.chomp
        if second=="\look"
                find
        end
        if $streets.include?(second)
                break
        end
end


nodeSecond = find_intersection first, second

puts "\n\n\n\n"

from = nodeFirst
to=nodeSecond



limit=1
retval2=false

$current=Array.new 0
$path=Array.new 0
$current << from
$path << $current

for go in 0..5  #Keeps going until route is found
        retval2=goRom from, to, limit
        if retval2==false
                limit+=1
                redo
        else
                break
        end
end

puts "Path is #{$current.each{|w| puts "#{w}, "}}"
puts "#{$current.each{|w| puts $interHash[w]}}\n"
```