class LinkedListNode

attr\_accessor :value, :next\_node

def initialize(value, next\_node=nil)

@value = value

@next\_node = next\_node

end

end

node1 = LinkedListNode.new(37)

node2 = LinkedListNode.new(99, node1)

node3 = LinkedListNode.new(12, node2)

def print\_values(list\_node)

if list\_node

print "#{list\_node.value} --> "

print\_values(list\_node.next\_node)

else

print "nil\n"

return

end

end

class Stack

attr\_reader :data

def initialize

@data = nil

end

# Push a value onto the stack

def push(value)

# IMPLEMENT ME!

if data.nil?

@data = LinkedListNode.new(value)

else

@data = LinkedListNode.new(value, @data)

end

end

# Pop an item off the stack.

# Remove the last item that was pushed onto the

# stack and return the value to the user

def pop

# I RETURN A VALUE

if data.nil?

return

end

ret = @data.value

@data = @data.next\_node

print ret

end

end

def reverse\_list(list)

# ADD CODE HERE

stack = Stack.new

while list

stack.push(list.value)

list = list.next\_node

end

stack.data

# ADD CODE HERE

end

print\_values(node3)

puts "-------"

revlist = reverse\_list(node3)

print\_values(revlist)

stack = Stack.new

# Pushes (adds) the number 1 to the empty stack

stack.push(1)

# Pushes the number 2 to the TOP of the stack

stack.push(2)

# Pops (removes) the TOP number from the stack (2)

puts stack.pop

# Pops the remaining number from the stack (1)

puts stack.pop

puts stack.pop