# Chris Clemmons

# CSC310

# 1/27/19

# sets up our variables

finished = False

ln = []

# loop to read/accept input while finished is false

while not finished:

# accepts lines until told to stop with control-d

try:

line = input('Please enter text, press ctrl-d to stop')

ln.append(line)

# when it reaches end of the input, it prints it out in reversed order

except EOFError:

for j in range(len(ln)-1, -1, -1):

# prints out the lines

print(ln[j])

# sets boolean to true to end the loop

finished = True

# Problem 2

# function to check for a pair of numbers with an odd product

def distinct\_pair(nums):

# for loop to check all numbers in nums

# i checks the first number against all others before moving onto the next entry in the array

for i in range(0, len(nums)):

# k checks every number after i until there is a match or the loop ends

for k in range(i+1, len(nums)):

# checks i against k

if (nums[i]\*nums[k]) % 2 == 1:

return True

return False

nums1 = [1, 2, 2, 3, 4]

nums2 = [1, 2]

print(distinct\_pair(nums1))

print(distinct\_pair(nums2))

# Problem 3 function to check for permutations

def perm(p):

# checks if there is 1 or less item in array

if len(p) <= 1:

return p

# checks permutations

else:

# runs from i to length to return all permutations

for i in range(0, len(p)):

# checks and returns permutations

for m in perm(p[:i] + p[i+1:]):

return [p[i]] + m

# for loop to print permutations

for perms in perm(["1", "2", "3"]):

print(perms)

# variables for hamming dist

x = 1

y = 4

# function to check hamming distance

def hamming(n1, n2):

# sets up count for n1 and n2

count1 = 0

count2 = 0

# checks n1

while n1 >= 1:

n1 = n1/2

count1 += 1

# checks n2

while n2 >= 1:

n2 = n2/2

count2 += 1

# gets the hamming dist from count1 and count2

count = abs(count1-count2)

# returns dist

return count

# prints hamming dist

print(hamming(x, y))