Q-1> Find the angle MBC

from math import atan

from math import degrees

ab = int (input())

bc = int (input())

print(str(round(degrees(atan((ab/2) / (bc / 2))))) + "°")

Q-2> Word Order

from collections import Counter, OrderedDict

class WordCount (Counter, OrderedDict):

pass

d = WordCount(input() for \_ in range (int(input())))

print(len(d))

print(\*d.values())

# Q-3> Compress the String!

from itertools import groupby

s = input()

for key, item in groupby(s):

print((list(item).count(key),int(key)), end=' ')

#### Q-4> Triangle Quest 2

for i in range(1,int(raw\_input())+1):

print(((10\*\*i -1)//9)\*((10\*\*i -1)//9))

Q-5> Triangle Quest

for i in range(1,input()):

print ((10\*i -1)//9)

# Q-6> Validating Email Addresses With a Filter

import re

def fun(s):

# return True if s is a valid email, else return False

a = re.match(r'[a-zA-Z0-9\_-]+@[a-zA-Z0-9]+\.[a-zA-Z]{1,3}$',s)

return(a)

def filter\_mail(emails):

return filter(fun, emails)

if \_\_name\_\_ == '\_\_main\_\_':

n = int(raw\_input())

emails = []

for \_ in range(n):

emails.append(raw\_input())

filtered\_emails = filter\_mail(emails)

filtered\_emails.sort()

print filtered\_emails

# Q-7>Reduce Function

from \_\_future\_\_ import print\_function

from fractions import Fraction

def product(fracs):

t =reduce(lambda a, b: a \* b, fracs) # complete this line with a reduce statement

return t.numerator, t.denominator

if \_\_name\_\_ == '\_\_main\_\_':

fracs = []

for \_ in range(input()):

fracs.append(Fraction(\*map(int, raw\_input().split())))

result = product(fracs)

print(\*result)

# Q-8>Validating Credit Card Numbers

# Enter your code here. Read input from STDIN. Print output to STDOUT

import re

pattern = r'(?!\d\*(\d)(-?\1){3})^[456][0-9]{3}(-?[0-9]{4}){3}$'

for i in range(int(input())):

m = re.match(pattern,input().strip())

if m:

print ('Valid')

else:

print ('Invalid')

# Q-9> Words Score

def is\_vowel(letter):

return letter in ['a', 'e', 'i', 'o', 'u', 'y']

def score\_words(words):

score = 0

for word in words:

num\_vowels = 0

for letter in word:

if is\_vowel(letter):

num\_vowels += 1

if num\_vowels % 2 == 0:

score += 2

else:

score += 1

return score

# Q-10>Default Arguments

class EvenStream(object):

def \_\_init\_\_(self):

self.current = 0

def get\_next(self):

to\_return = self.current

self.current += 2

return to\_return

class OddStream(object):

def \_\_init\_\_(self):

self.current = 1

def get\_next(self):

to\_return = self.current

self.current += 2

return to\_return

def print\_from\_stream(n, stream=EvenStream()):

stream.\_\_init\_\_()

for \_ in range(n):

print(stream.get\_next())

queries = int(input())

for \_ in range(queries):

stream\_name, n = input().split()

n = int(n)

if stream\_name == "even":

print\_from\_stream(n)

else:

print\_from\_stream(n, OddStream())