Họ tên:        LÊ HOÀNG ANH TÚ

1. Giai thừa của 1 số nguyên dương n là tích của tất cả các số nguyên dương nhỏ hơn hoặc bằng n. Công thức như sau:

https://lh5.googleusercontent.com/Vcz5869pR5c493vU6CA4DULDCBsQWxlzqQtsMW06nuZSiIwFfHq4X9hbwRB6rXKCj0kF-HDiFR2E-3vGjo-I99AUvrg7RBur_6HiPTLDT7aOAYeO4FBCrEYafgf-CtFqvrS4YIxC

Viết mã giả mô tả thuật toán tính giai thừa của một số n cho trước.

def giai\_thua(n):

if n < 0 or type(n) is not int :

print ("n khong phai so nguyen duong , nhap lai n")

else :

result = 1

for i in range(1,n+1):

result = result \*i

print ("{}! = {}".format(n,result))

giai\_thua(10)

2. Cho định nghĩa sau: số hoàn hảo là số có giá trị bằng tổng các ước nhỏ hơn nó.

Ví dụ: 6=1+2+3; 28=1+2+4+7+14.

Viết mã giả mô tả thuật toán kiểm tra 1 số n cho trước có phải là số hoàn hảo hay không.

def perfect\_number(n):

if (n <= 0) or ( type(n) is not int) :

print (" {} khong phai so nguyen duong duong, moi nhap lai ! ".format(n))

else:

list\_uoc = []

for i in range(1,n-1):

if n % i == 0 :

list\_uoc.append(i)

sum\_uoc = 0

for j in list\_uoc:

sum\_uoc += j

if sum\_uoc == n :

print (" {} la so hoan hao ! ".format(n))

if sum\_uoc != n :

print (" {} khong phai so hoan hao, try again :3 ".format(n))

perfect\_number(28)

3. Với chuỗi cho trước có độ dài lớn hơn 1, viết mã giả thể hiện thuật toán xác định vị trí của 1 kí tự trong chuỗi sao cho khi xoá kí tự đó đi thì chuỗi còn lại là nhỏ nhất có thể.

Ví dụ: chuỗi `231` sau khi xoá đi 1 kí tự có thể trở thành `31`, `21`, `23`. Chuỗi nhỏ nhất là `21`, đáp án đúng là xoá đi kí tự `3` trong chuỗi ban đầu.

def tim\_vi\_tri\_can\_xoa(n):

if len(str(n)) == 1 :

print ("chuoi co do dai nho hon 1, moi nhap lai ! ")

else :

do\_dai = len(str(n))

list\_cutted = []

for i in range(0,do\_dai):

n\_list = list(str(n))

n\_list.pop(i)

new\_str = "".join(n\_list)

list\_cutted.append(int(new\_str))

smallest\_num = list\_cutted[0]

for num in list\_cutted:

if smallest\_num >= num:

smallest\_num = num

index = list\_cutted.index(num) +1

deleted\_number = str(n)[index-1]

print('Xoa ki tu co vi tri thu {} - tuong ung so "{}" de duoc so nho nhat co gia tri {}'.format(index,deleted\_number,smallest\_num))

tim\_vi\_tri\_can\_xoa(231)

4. Bob cầm **n** đồng đi mua kẹo. Giá của 1 viên kẹo là **c** đồng. Cửa hàng lại có chương trình khuyến mại cứ **m** tờ giấy gói kẹo thì đổi được 1 viên kẹo mới. Viết mã giả thể hiện thuật toán xác định tổng số kẹo Bob có thể mua được từ các tham số **n**, **c**, **m** như đã mô tả ở trên.

Ví dụ:

* n=10, c=2, m=5: Bob mua được 5 viên kẹo, sau đó đổi 5 tờ giấy gói lấy 1 viên kẹo nữa, tổng cộng đáp án là 6.
* n=12, c=4, m=4: Bob mua được 3 viên kẹo, số giấy gói kẹo không đủ đổi thêm nữa, đáp án cuối cùng là 3.
* n=6, c=2, m=2: Bob mua được 3 viên kẹo, lấy 2 trong 3 tờ giấy gói kẹo đổi được thêm 1 viên. Sau đó dùng tiếp 1 tờ giấy gói dư ở lần đổi thứ nhất + tờ giấy gói của viên kẹo ở lần đổi thứ 2, đổi được thêm 1 viên kẹo nữa. Đáp án tổng cộng là 5.

def bob\_can\_buy(n,c,m):

if n <= 0 or c <= 0 or m <=0 :

print ("Wrong Input")

if n < c :

print ("Bob khong du tien mua 1 vien keo rui :( xin me them nhe ! ")

if n == c :

print ("Bob du tien mua 1 vien keo duy nhat, enjoy ~!")

if n > c and m == 1 :

print ("Bob an keo forever :D ")

if n > c and m >1 :

candy\_can\_buy = int(n/c)

candy\_can\_trade = int(candy\_can\_buy/m)

leftover\_wrappers = candy\_can\_buy % m

if (candy\_can\_trade + leftover\_wrappers) < m :

total\_candy = candy\_can\_buy + candy\_can\_trade

if (candy\_can\_trade + leftover\_wrappers ) >= m :

total\_wrappers = leftover\_wrappers + candy\_can\_trade

candy\_can\_trade += int(total\_wrappers/m)

total\_candy = candy\_can\_buy + candy\_can\_trade

print ("Bob co the mua {} vien keo".format(total\_candy) )

bob\_can\_buy(10,2,2)