*#1a*

def factorial(x):

if x==1:

return x

else:

return(x\*factorial(x-1))

print(factorial(3))

In [ ]:

*#1b*

def fibonacci(x):

if x<=1:

return x

else:

return fibonacci(x-1)+fibonacci(x-2)

print(fibonacci(8))

In [ ]:

*#1c*

def arrayPrint(arr):

n=len(arr)

if n==0:

return

else:

print(arr[0])

arrayPrint(arr[1:])

arrayPrint([1,2,3,4,5,6,8,7])

In [ ]:

*#1d*

def powerN(x,n):

if n==1:

return x

else:

return x\*powerN(x,(n-1))

print(powerN(3,3))

In [ ]:

*#2b*

class Node:

def \_\_init\_\_(self, e, n):

self.element = e

self.next = n

class LinkedList:

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

self.head = None

tail = self.head

if type(a) != Node:

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

if i==0:

self.head=Node(a[i],None)

tail=self.head

else:

temp = Node(a[i],None)

tail.next = temp

tail = tail.next

else:

self.head=a

def printer(arr):

if arr==None:

return 0

else:

return arr.element+printer(arr.next)

a=[1,2,3,4]

Li=LinkedList(a)

print(printer(Li.head))

In [ ]:

*#2c*

class Node:

def \_\_init\_\_(self, e, n):

self.element = e

self.next = n

class LinkedList:

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

self.head = None

tail = self.head

if type(a) != Node:

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

if i==0:

self.head=Node(a[i],None)

tail=self.head

else:

temp = Node(a[i],None)

tail.next = temp

tail = tail.next

else:

self.head=a

def printer(arr):

if arr==None:

return None

else:

printer(arr.next)

print(arr.element)

a=[10,20,30,40]

Li=LinkedList(a)

printer(Li.head)

*#3*

def hocBuilder(floor):

if floor==0:

return 8

else:

return 5+hocBuilder(floor-1)

print(hocBuilder(2))

In [ ]:

*#4a*

def stairs(n):

if n==0:

return ''

else:

x=(stairs(n-1))+str(n)

print(x)

return x

x=stairs(5)