Ramya J

3BR22EC131

```
#1.FACTORIAL
fact=1
n=int(input('enter a num:'))
for i in range(n,0,-1):
    fact=fact*i
print(fact)
```

```
enter a num:5
120
```

```
##2.FIBINOCI
n=int(input('enter a number:'))
a=0
b=1
i=0
while i<n:
    c=a+b
    a=b
    b=c
    i+=1
    print(a)
```

```
enter a number:5
1
1
2
3
5
```

```
##3.ARMSTRONG
def armstrong():
    num=int(input('enter a number:'))
    arms=str(num)
    result = 0
    for i in arms:
        result=result+int(i)**len(arms)
    print(result)
    if result==num:
        print('armstrong')
    else:
        print('not armstrong')
armstrong()
```

```
enter a number:153
153
armstrong
```

```
##4.REVERSE INTEGER
l = []
n =int(input('Enter the num of elements:'))
for i in range(n):
    l.append(int(input('enter the elements')))
print('list=',l)
print('Reveresed list=',l[::-1])
```

```
Enter the num of elements:4
enter the elements1
enter the elements2
enter the elements3
enter the elements4
list= [1, 2, 3, 4]
Reveresed list= [4, 3, 2, 1]
```

```
##5.RECUSSION FACTORIAL
def cal_fac(n):
    fact = 1
    for i in range(1,n+1):
        fact = fact * i
    print(fact)
cal_fac(5)
```

```
120
```

```
##6.RIGHT SIDE TRIANGLE UPWARD
def pattern():
    n=5
    for i in range(1,n+1):
        print(' *'*i)
pattern()
```

```
 *
 * *
 * * *
 * * * *
 * * * * *
```

```
##7.RIGHT SIDE TRIANGLE DOWNWARD
def pattern1():
    n=5
    for i in range(n , 0, -1):
          print(' *'*i)
pattern1()
```

```
 * * * * *
 * * * *
 * * *
 * *
 *
```

```
##8.LEFT SIDE TRIANGLE UPWARD
def pattern3():
    n=5
    for i in range(1,n+1):
        print(' '*(n-i),'*'*i)
pattern3()
     *
    **
   ***
  ****
 *****

##9.LEFT SIDE TRIANGLE DOWNWARD
def pattern4():
  n=5
  for i in range(1,n+1):
        print(' '*i,'*'*(n-i))
pattern4()
  ****
   ***
    **
     *


##10.USER NAME PASSWORD
name=input('Enter a name:')
if name=='ramya':
    print('hi, ramya')
    pas=input('Enter a password:')
    if pas=='123':
        print('Access granted')
    while pas!='123':
        new_pas=input('Enter a new password')
        if new_pas=='123':
            print("Access granted")
            break
else:
    print('name is not found')
Enter a name:ramya
hi,ramya
```

```
Enter a password:123
Access granted

##11.USER NAME PASSWORD WITH ATTEMPTS
u_n=input('Enter a name:')
if u_n=='ramya':
    print('hi,ramya')
    for i in range(1,4):
        u_p=input("enter a password")
        if u_p=='123':
            print('welcome')
            break
        else:
            print(f"{i}attempts done/n {3-
i} attempts are remaining")
        if i==3:
            print('Account block')
else:
    print('invalid user')
Enter a name:ramya
hi,ramya
enter a password12
1attempts done/n 2 attempts are remaining
enter a password1
2attempts done/n 1 attempts are remaining
enter a password2
3attempts done/n 0 attempts are remaining
Account block

##12.ATM
name=['ramya','lavanya','poojitha','bhanu']
pas=[1,2,3,4]
balance=[1000,1500,1200,1000]


def withdraw(current):
    amt=int(input('Enter a amount:'))
    if amt<=balance[current]:
        balance[current]-=amt
        print(balance[current])
    else:print('insufficient')
def deposit(current):
    amt=amt=int(input('Enter a amount:'))
    balance[current]+= amt
    print(balance[current])
def c_balance(current):
    print('Balance is:',balance[current])
def default(current):
    print("enter crt option")
```

```
u_n=input('Enter a name:')
u_p=int(input('Enter a password:'))
for i in range(len(name)):
    if u_n==name[i]:
        print('hello')
        if u_p==pas[i]:
            while True:
                print('1:withdraw\n 2:deposit\n 3:balance')
                option=int(input('Enter option:'))
                if option==0:break
                data={1:withdraw,2:deposit,3:c_balance}
                res=data.get(option,default)
                res(i)
```

```
Enter a name:ramya
Enter a password:1
hello
1:withdraw
 2:deposit
 3:balance
Enter option:2
Enter a amount:1000
2000
1:withdraw
 2:deposit
 3:balance
Enter option:3
Balance is: 2000
1:withdraw
 2:deposit
 3:balance
Enter option:0
```

```
##13.SQUARE PATTERN
n=int(input('Enter a value:'))
for i in range(1,n+1):
    if i==1 or i==n:
        print('* '*n)
    else:print('* ','  '*(n-3),'*')
```

```
Enter a value:4
* * * *
*     *
*     *
* * * *
```

```
##14.UPWARD TRIANGLE
def pattern6():
    n = int(input('Enter a num:'))
    for i  in range(1,n+1):
        print(' '*(n-i),'*'*(i+(i-1)))
pattern6()
```

```
Enter a num:5
     *
    ***
   *****
  *******
 *********
```

```
##15.DOWNWORD TRIANGLE
def pattern7():
    n = int(input('Enter a num:'))
    for i in range(n,0,-1):
        print(' '*(n-i),'*' *(i+(i-1)))
pattern7()
```

```
Enter a num:5
 *********
  *******
   *****
    ***
     *
```

```
##16.WORD PATTERN
def name1():
    w=input('Enter a word:')
    for i in range(len(w)):
        print(w[0:i+1])
    for i in range(len(w), 1, -1):
        print(w[0:i - 1])
name1()
```

```
Enter a word:RAMYA
R
RA
RAM
RAMY
RAMYA
RAMY
RAM
```

```
RA
R
```

##COMBINATIONS

```
from itertools import *
l=[1,2,10,3,4,5,6,7,8,9,0]
print(list(combinations(l,2)))
```

```
##ROTATION
name=input('Enter a word:')
n=int(input('Enter 0 or 1:'))
if n==0:
    num=int(input('Enter num of times to be rotated:'))
    for i in range(num):
        name=name[-1] + name[:-1]
    print(name)
    # break
elif n==1:
    num = int(input('Enter num of times to be rotated:'))
    for j in range(num):
        name=name[1:] + name[0]
    print(name)
        # break
else:
    print('Enter 0 or 1')
```

```
##COUNT
def count(na):
    if na==0:
        return
    else:
        print(na)
        return count(na-1)
count(10)
```

```
##LINEAR SEARCH

l=list(map(int,input('Enter a list values:').split(' ')))
key=int(input('enter value to be find:'))
for i in range(len(l)):
    if l[i]==key:
        print(f"the value is found at {i} index")
```

```
        break
if key not in l:
    print('value not found')
if i==len(l)-1 and l[i]!=key:
    print('value not found')
```

```
##BINARY SEARCH

def b_s(l,k):
    beg=0
    end=len(l)-1
    while beg<=end:
        mid=(beg+end)//2
        if l[mid]==k:
            print(f"value is found at index {mid}")
            break
        elif l[mid]>k:
            end=mid-1
        else:
            beg=mid+1
    if k not in l:
        print('value not found')
l=[1,2,3,4,5,6,7,8,9,10]
k=int(input('enter a value to be find:'))
b_s(l,k)
```

```
##MATPLOTLIB

from matplotlib import pyplot
a=[1,2,3,4,5]
b=[2,5,4,3,6]
pyplot.bar(a,b,width=0.25)
pyplot.plot(a,b)
pyplot.show()
```

```
##BUBBLE SORT
l=[2,6,7,3,8,7,1]
print('original list:',l)
n=len(l)
for i in range(n):
    for j in range(n-i-1):
        if l[j]>l[j+1]:
            l[j],l[j+1]=l[j+1],l[j]
print('sorted list:',l)
```

```
##selection sort
l=[2,7,6,3,8,1]
print('original list:',l)
n=len(l)
for i in range(n):
    min=i
    for j in range(i+1,n):
        if l[min]>l[j]:
            min=j
    l[i],l[min]=l[min],l[i]
print('sorted list:',l)


##INSERTION SORT
l=list(map(int,input("enter a values:").split(' ')))
for i in range(1,len(l)):
    key=l[i]
    j=i-1
    while j>=0 and l[j]>key:
        l[j+1]=l[j]
        j=j-1
    l[j+1]=key
print(l)

## MERGE SORT
def merge(lis):
    if len(lis)>1:
        mid=len(lis)//2
        left=lis[:mid]
        right=lis[mid:]
        merge(left)
        merge(right)
        l=0
        r=0
        k=0
        while l<len(left) and r<len(right):
            if left[l]>right[r]:
                lis[k]=right[r]
                r=r+1
            else:
                lis[k]=left[l]
                l=l+1
            k=k+1
        while l<len(left):
            lis[k]=left[l]
            l=l+1
            k=k+1
        while r<len(right):
            lis[k]=right[r]
            r=r+1
            k=k+1
```

```
l=[4,3,8,7,1,2]
print('Original list:',l)
merge(l)
print('Sorted list:',l)


##SINGLE LINKED LIST
class node:
    def __init__(self,data):
        self.data=data
        self.next=None

class linkedlist:
    def __init__(self):
        self.head=None

    def insert_at_end(self,data):
        newNode=node(data)
        if self.head is None:
            self.head=newNode
        else:
            current=self.head
            while current.next is not None:
                current=current.next
            current.next=newNode

    def insert_at_beg(self,data):
        newNode=node(data)
        if self.head is None:
            self.head=newNode
        else:
            newNode.next=self.head
            self.head=newNode

    def insert_at_position(self,data,position):
        newNode = node(data)
        if self.head is None:
            self.head = newNode
        elif position==1:
            self.head=newNode
        else:
            current=self.head
            count=0
            while current:
                count+=1
                if count==position-1:
                    newNode.next=current.next
                    current.next=newNode
                    break
                current=current.next
```

```
            if count<position-1:
                self.insert_at_end(data)

    def delete_at_beg(self):
        if self.head is None:
            print('no data found')
        else:
            self.head=self.head.next
            self.head.next=None


    def delete_at_end(self):
        if self.head is None:
            print('no data found')
        else:
            current=self.head
            while current.next.next:
                current=current.next
            current.next=None

    def delete_by_value(self,data):
        if self.head is None:
            print('no data found')
        elif self.head.data==data:
            self.delete_at_beg()
        else:
            current=self.head
            count=0
            while current.next:
                if current.next.data==data:
                    count+=1
                    current.next=current.next.next
                    break
            if count==0:
                print('no data found')

    def search_by_value(self,key):
        if self.head is None:
            print('no data found')
        else:
            current=self.head
            count=0
            while current:
                count += 1
                if current.data==key:
                    #
                    print('data is found at position',count)
                    count =-1
                    break
                current=current.next
            if count!=-1:
```

```
                print('data not found')


    def display(self):
        current=self.head
        while current:
            print(current.data,end="-->")
            current=current.next


obj=linkedlist()
obj.insert_at_end(5)
obj.insert_at_beg(10)
obj.insert_at_beg(3)
obj.insert_at_beg(7)
obj.display()


obj.insert_at_end(6)
obj.insert_at_position(2,3)
obj.delete_at_beg()
obj.insert_at_beg(5)
obj.delete_at_beg()



##DOUBLE LINKED LIST

class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
        self.previous=None
class double_link_list:
    def __init__(self):
        self.head=None
        self.tail=None
    def insert_at_end(self,data):
        newNode=Node(data)
        self.tail=newNode
        if self.head is None:
            self.head=newNode
        else:
            current=self.head
            while current.next:
                current=current.next
            current.next=newNode
            newNode.previous=current
    def insert_at_beg(self,data):
        newNode=Node(data)
```

```
            self.tail=newNode
            if self.head==None:
                self.head=newNode
            else:
                newNode.next=self.head
                self.head=newNode
                current=newNode.previous

        def display_f(self):
            current = self.head
            while current:
                print(current.data,end="-->")
                current=current.next
        def display_b(self):
            current=self.tail
            while current:
                print(current.data,end='<--')
                current=current.previous
obj=double_link_list()
# obj.insert_at_end(5)
obj.insert_at_beg(1)
# obj.insert_at_end(4)
obj.insert_at_beg(6)
# obj.insert_at_end(3)
obj.display_f()
print()
obj.display_b()


##STACK
class stack:
    def __init__(self):
        self.stack=[]
        self.size=4
        self.top=-1
    def push(self,item):
        if self.top<self.size-1:
            self.stack.append(item)
            self.top=self.top+1
        else:
            print('Overflow')
    def pop(self):
        if self.isEmpty():
            print('Underflow')
        else:
            self.stack.pop()
            self.top=self.top-1
    def isEmpty(self):
        if self.top==-1:
            return True
        else:
```

```
            return  False

    def display(self):
        for i in range(len(self.stack)-1,-1,-1):
            print(self.stack[i])
obj=stack()
obj.push('ramya')
obj.push('lavanya')
obj.push('bhanu')
obj.push('poojitha')
obj.push('1')
obj.pop()
obj.pop()
obj.pop()
obj.pop()
obj.pop()
obj.display()

##FILE HANDLING
with open('mindmap.png','rb') as file:
    data=file.read()
with open('image.jpg','wb') as file1:
    file1.write(data)

##FREQUENCY
st='I like chocolate and drink . my bro also like chocolate
and drink .'
dict_words={}
for line in st:
    words=line.split()
    for word in words:
        dict_words[word]=dict_words.get(word,0)+1
list_words=[]
for key,val in dict_words.items():
    list_words.append((val,key))

print(list_words)



##FIND THE LARGEST
a=int(input('enter a number:'))
b=int(input('enter a number:'))
c=int(input('enter a number:'))
lis=[a,b,c]
lis.sort(reverse=True)
print(lis)
print("largest=",lis[0])
print("sec_lar=",lis[1])
print("third_lar=",lis[2])
#if condition
```

```
if a>b and a>c:
    print('a 1')
    if b>c:
        print("b 2\nc 3")
    else:
        print('c 2\nb 2')
elif b>a and b>c:
    print('b 1')
    if a>c:
        print('a 2\nc 3')
    else:
        print('c 2\na 3')
else:
    print('c 1')
    if a>b:
        print('a 2\nb 3')
    else:
        print('b 2\na 3')



##CLASS
class Student:
    def __init__(self,marks):
        self.marks=marks
        self.total()
    def total(self):
        self.total=0
        for i in  self.marks:
            self.total=self.total+i
        print(self.total)
    def rank(*self):
        l=[]
        for i in self:
            l.append(i.total)
        print(l)
        sort=sorted(l)
        print(l)


ramya=Student([100,95,97,92,95])
roja=Student([95,96,97,92,98])
Student.rank(roja,ramya)



##ABSTRACT METHOD
from abc  import ABC,abstractmethod
class printable(ABC):
    @abstractmethod
    def print_content(self):
```

```
        pass
class document(printable):

    def print_content(self):
        print('hello')
doc=document()
doc.print_content()

##OPERATOR OVERLOADING
class A:
    def __init__(self,a):
        self.a=a
    def __truediv__(ramya,roja):
        return (ramya.a/roja.a)
roja=A(10)
ramya=A(20)
print(ramya/roja)
print(10*10)


##
class ramya:
    def fav_food(self):
        print('masala dosa')
class lav:
    def fav_food(self):
        print('annam pappu')
class pooji:
    def fav_food(self):
        print('plain dosa')
class bhanu:
    def fav_food(self):
        print('annam muddhapappu')
def fav_food(food):
    food.fav_food()
r=ramya()
l=lav()
b=bhanu()
p=pooji()
fav_food(l)



#QUADRATIC EQUATIONS
import cmath,math
a=1
b=5
c=6
d=((b**2)-(4*a*c))
sol1=((-b-cmath.sqrt(d))/(2*a))
```

```
sol2=((-b+cmath.sqrt(d))/(2*a))
print('two solutions are:',sol1,sol2)


#RANDOM
import random
print(random.randint(10,20))
lis=['ramya','lavanya','poojitha','bhanu']
print(random.choice(lis))
##prime number
a=int(input('Enter a num:'))
if a==1:
    print('1 is not a pm')
elif a>1:
    for i in range(2,a):
        if a%i==0:
            print('not pm')
        else:
            print('pm')
        break
else:
    print('enter positive num')

for num in range(1,11):
    if num>1:
        for i in range(2,num):
            if num%i==0:
               break
        else:
            print(num)


## LCM
def lcm(a,b):
    if a>b:
        g=a
        print(a,'is greater')
    else:
        g=b
        print(b,'is greater')
    while(True):

        if (g%a==0) and (g%b==0):
            res=g
            break
        g+=1
    return res
print('The lcm is',lcm(54,24))
```

```
## factors
def factors(x):
    print('the factors of',x,'are:')
    for i in range(1,x+1):
        if x%i==0:
            print(i)

print(factors(10))


## tables
n=int(input('Enter a num:'))
for i in range(1,11):
    print(n,'*',i,'=',n*i)


##calender
import calendar
yy=int(input('Enter a year:'))
mm=int(input('enter a month:'))
print(calendar.month(yy,mm))


##CLASS AND OBJECT
lass student:
    clg_name='BITM'
    def __init__(self,name,usn,branch):
        self.name=name
        self.usn=usn
        self.branch=branch
    def display(self,name,usn,branch):

print('STUDENT:','\n',self.name,'\n',self.usn,'\n',self.branch
,'\n',self.clg_name)
s= student('ramya','3br22ec131','ECE')
s.display('ramya','3br22ec131','ECE')


##
class food:
    def __init__(self,*snacks):
        self.snacks1='pizza'
        self.snacks2 = 'samosa'
        self.snacks3 = 'cake'
    def display(self):

print('snacks:',self.snacks1,self.snacks2,self.snacks3)
items=food('pizza','samosa','cake')
items.display()
```

```
##
import math
class circle:
    def __init__(self,radius):
        self.radius=radius
    def perimeter(self):
        print('perimeter=',2*(math.pi)*self.radius)
    def area(self):
        print('area=',(math.pi)*(self.radius**2))
c=circle(5)
c.perimeter()
c.area()


##AGE
from datetime import date
class person:
    def __init__(self,name,country,dob):
        self.name=name
        self.country=country
        self.dob=dob
    def c_age(self):

        today=date.today()
        age=today.year-self.dob.year
        if
today<(date(today.year,self.dob.month,self.dob.day)):
            age-=1
        return age
    def display(self):

print('name=',self.name,'\ncountry=',self.country,'\ndate of
birth=',self.dob,'\nage=',self.c_age())
person1=person('ramya','INDIA',date(2004,12,10))
person2=person('roja','INDIA',date(2008,6,10))
person3=person('bhanu','INDIA',date(2004,7,12))
person4=person('Lavanya','INDIA',date(2004,11,18))
person5=person('Poojitha','INDIA',date(2004,4,30))
person1.c_age()
person2.c_age()
person3.c_age()
person4.c_age()
person5.c_age()
person5.display()


##SWITCH
def switch_case():

    while n!=0:
        v=input('Enter yes to continue:')
```

```
        if v=='yes':

switch_data={1:armstrong,2:arm_lambda,3:pattern1,4:pattern2}
            res=switch_data.get(n,default)
        res()
n=int(input('Enter a function name:'))
```

```
##try and except method
a=2
b=0
try:
    print(a/b)
except Exception as e:
    print(e)
else:
    print('nothing')
finally:
    print('done')
```