**OOPS LAB**

**-**Raajas Bhatt 23FE10CSE00153

**LAB 1**

n=int(input("Enter a number : "))  
flag=0  
for i in range(2,n):  
 if (n%i==0):  
 flag=1  
 break  
if (flag==0):  
 print("prime number")  
else :  
 print("composite number")  
  
n=int(input("Enter a number : "))  
reversed\_num=0  
test\_num=n  
while (test\_num != 0) :  
 digit = test\_num%10  
 reversed\_num = reversed\_num\*10 + digit  
 test\_num= test\_num//10  
print(reversed\_num)  
if (n==reversed\_num):  
 print("Palindrome")  
else :  
 print("Not a Palindrome")  
  
p=int(input("Enter percentage : "))  
if (100<=p and p>=91) :  
 print("Grade : A+")  
elif (p>=81) :  
 print("Grade : A")  
elif (p>=71) :  
 print("Grade : B+")  
elif (p>=61) :  
 print("Grade : B")  
elif (p>=51) :  
 print("Grade : C")  
elif (p>=41) :  
 print("Grade : D")  
elif (p>=35) :  
 print("Grade : E")  
else :  
 print("Grade : F")  
  
import math  
n=int(input("Enter a number : "))  
s=0  
t=n  
while(t != 0):  
 digit = t%10  
 s=s+ math.pow(digit,3)  
 t //= 10  
print(s)  
if (n==s):  
 print("Armstrong Number")  
else :  
 print("Not an armstrong number")  
  
  
import math  
r=int(input("Enter radius : "))  
a= math.pi \* r \* r  
print("Area : ",a)  
  
  
n=int(input("Enter a number : "))  
l=[]  
for i in range (1,n):  
 s=i\*i  
 l.append(s)  
x=len(l)  
flag=0  
for j in range (0,x-1):  
 if (n==l[j]):  
 flag=1  
 break  
if (flag==1):  
 print("Perfect Square")  
else:  
 print("Not a Perfect Square")  
  
  
import math  
n=int(input("Enter a number : "))  
t=n  
s=0  
while(t != 0):  
 digit=t%10  
 s=s + math.factorial(digit)  
 t //= 10  
print(s)  
if (n==s):  
 print("Strong Number")  
else :  
 print("Not a strong number")

**LAB 2**

#fibbonaci series  
n=int (input("Enter the number of terms in the Fibbonacci series : "))  
a1=0  
a2=1  
print(a1,a2,end=" ")  
for i in range(0,n-2):  
 a3=a1+a2  
 print(a3,end=" ")  
 a1=a2  
 a2=a3  
  
#factorial using loops  
n=int (input("Enter number whose multiplication table is to be printed : "))  
for i in range (1,11):  
 print(n,"\*",i,"=",n\*i)  
  
# gcd and lcm using loops  
  
def greater(m,n):  
 if (m>n):  
 greater=m  
 else :  
 greater=n  
 return greater  
  
n=int (input("Enter first number : "))  
m=int(input("Enter second number : "))  
  
for i in range (greater(m,n),0,-1):  
 if (n%i==0 and m%i==0) :  
 gcd=i  
 break  
  
x=greater(n,m)  
while (True):  
 if (x%n==0 and x%m==0):  
 lcm=x  
 break  
 else:  
 x +=1  
  
print("gcd : ",gcd)  
print("lcm : ",lcm)

**LAB 3**

# Disarium Number  
import math  
n=int(input("Enter a number : "))  
temp=n  
temp1=n  
sum=0  
  
c=0  
while(temp1!=0):  
 temp1 //= 10  
 c += 1  
  
i=c  
while (temp!=0):  
 digit = temp%10  
 sum += math.pow(digit,i)  
 print(digit)  
 print(i)  
 print(sum)  
 temp //= 10  
 i -= 1  
  
if (sum==n): print("Disarium No.")  
else : print("Not a disarium number")  
  
  
# Harshad Number  
n=int(input("Enter a number : "))  
temp=n  
sum=0  
  
while(temp!=0):  
 digit = temp%10  
 sum += digit  
 temp //= 10  
  
if (n%sum==0): print("Harshad No.")  
else : print("Not a Harshad number")  
  
  
  
# Armstrong numbers from 1 to 1000  
import math  
  
for i in range (1,1001):  
 sum=0  
 temp =i;  
 while(temp != 0):  
 digit = temp%10  
 sum += math.pow(digit,3)  
 temp //= 10  
 if (sum==i): print(i,end=" ")

**LAB 4**

#LAB 04  
#Pattern Printing, Strings  
  
*"""1"""*"""1 2"""  
"""1 2 3"""  
"""1 2 3 4"""  
"""1 2 3 4 5"""  
  
for i in range (1,6):  
 for j in range (1,i+1):  
 print(j,end=" ")  
 print("")  
  
  
"""1"""  
"""2 2"""  
"""3 3 3"""  
"""4 4 4 4"""  
"""5 5 5 5 5"""  
  
for i in range (1,6):  
 for j in range (1,i+1):  
 print(i,end=" ")  
 print("")  
  
  
"""A"""  
"""B B"""  
"""C C C"""  
"""D D D D"""  
"""E E E E E"""  
  
for i in range (1,6):  
 for j in range (1,i+1):  
 print(chr(i+64),end=" ")  
 print("")  
  
  
"""\*\*\*\*\*"""  
"""\*\*\*\*"""  
"""\*\*\*"""  
"""\*\*"""  
"""\* """  
  
for i in range (1,6):  
 for j in range (i,6):  
 print("\*",end=" ")  
 print("")  
  
  
"""1 2 3 4 5"""  
"""1 2 3 4"""  
"""1 2 3"""  
"""1 2"""  
"""1 """  
for i in range (1,6):  
 k=1;  
 for j in range (i,6):  
 print(k,end=" ")  
 k+=1  
 print("")  
  
  
""" 1   
 121  
 12321  
 1234321  
123454321"""  
  
n=int(input("Enter n : "))  
for i in range (1,n+1):  
 k=1  
for j in range (i,n):  
 print(" ",end="")  
 for j in range (1,i+1):  
 print(k,end="")  
 k+=1  
 x=k-1  
 for j in range (1,i):  
 x-=1  
 print(x,end="")  
print("")  
  
  
# String  
# 1. Write a program to separate the following string into comma (,) separated values.  
# X= “ India.is.my.country”  
# 2. Write a program to sort strings alphabetically in python.  
# 3. Write a program to remove a given character from a string.  
# Y=”M.A.N.I.P.A.L”  
# 4. Write a program to remove the (.) dots from the above string.  
# 5. Write a program to take an input from a user as a string then, reverse it.  
# 7. Write a program to check if a string contains only digits.  
# 8. Write a program to check if a string is palindrome or not?  
# 9. Write a program to find the number of vowels in the string.  
# 10. Write a program to check if every word in a string begins with a capital letters or not.  
  
#Q1  
x= "India.is.my.country"  
z=x.replace(".",",")  
print(z)  
  
#Q2  
y="manipal"  
X=list(y)  
X.sort()  
y=str(X)  
print(y)  
  
#Q3  
a="manipal"  
b=list(a)  
b.remove("a")  
for i in b:  
 print(str(i),end="")  
  
#Q4  
print("")  
Y="M.A.N.I.P.A.L"  
print(Y.replace(".",""))  
  
#Q5  
for i in range (len(Y)-1,-1,-1):  
 print(Y[i],end="")  
  
#Q7  
print("")  
if Y.isdigit():  
 print("Yes")  
else:  
 print("NO")  
  
#Q8.  
Y="ABA"  
Y1=Y[::-1]  
if (Y==Y1):  
 print("Yes")  
else:  
 print("NO")  
  
#Q9.  
vowels=['a','e','i','o','u']  
c=0  
sentence="hello how are you"  
for word in sentence:  
 for alphabet in word:  
 if alphabet in vowels:  
 c +=1  
print(c)  
  
#Q10.  
sentence="hello how are you"  
if (sentence[0][0][0]).isupper():  
 print("Yes")  
else:  
 print("NO")

**LAB 5**

#LAB 05  
#Lambda, Map(), Reduce(), Filter()  
  
#Q1  
triple=lambda a : a\*3  
l1=[1,2,3,4,5]  
l2=map(triple,l1)  
print(list(l2))  
  
#Q2  
add4=lambda a,b,c,d : a+b+c+d  
l1=[1,2,3,4,5]  
l2=[1,2,3,4,5]  
l3=[1,2,3,4,5]  
l4=[1,2,3,4,5]  
l5=map(add4,l1,l2,l3,l4)  
print(list(l5))  
  
#Q3  
l1=["hello","how","are","you"]  
l2=map(list,l1)  
print(list(l2))  
  
#Q4  
import math  
l1=[0,1,2,3,4]  
l2=map((lambda a,n : math.pow(a,n)),l1,range(len(l1)))  
print(list(l2))  
  
#Q5  
l1=[1,2,3,4]  
square=map( (lambda a : a\*a), l1)  
print(list(square))  
  
#Q6  
l1=["H","E","e","l","L","o","o","H"]  
l1=set(l1)  
l2=filter(str.isupper,l1)  
l3=filter(str.islower,l1)  
print(list(l2))  
print(list(l3))  
  
#Q7  
l1=[5,6,7,8]  
l2=[1,2,3,4]  
l3=map((lambda a,b : a-b),l1,l2)  
print(list(l3))  
  
#Q8  
l=[5,6,7,8]  
t=(1,2,3,4)  
sl=map(str,l)  
st=map(str,t)  
print(list(sl))  
print(list(st))  
  
#Q9  
l1=(9,8,7,5,4,3,2)  
print(l1)  
l2=[]  
while(True):  
 x=int(input("Enter position of elements to convert into string : "))  
 l2.append(l1[x])  
 choice=input("Do you want to enter more positions to convert to string (yes/no) : ")  
 if (choice=="no"):  
 break  
l3=map(str,l2)  
print(list(l3))  
  
#Q10  
n=int(input("Enter n : "))  
a1=0  
a2=1  
print(a1," ",a2," ",end="")  
l=[a1,a2]  
for i in range (0,n-2):  
 a3=a1+a2  
 print(a3," ",end="")  
 l.append(a3)  
 a1=a2  
 a2=a3  
print("")  
l1=map((lambda a : a\*a ),l)  
print(list(l1))  
  
#Q11  
import functools  
arr=[1,2,3,4,5]  
sum=functools.reduce((lambda a,b : a+b),arr)  
print(sum)

**LAB 6**

#LAB 06  
#Classes and Objects  
  
#Q1  
class Person:  
 def \_\_init\_\_(self,name,age):  
 self.name=name  
 self.age=age  
  
p1=Person("Spriha",18)  
p2=Person("ABCD",19)  
  
print("Name : ",p1.name)  
print("Age :",p1.age)  
print("Name : ",p2.name)  
print("Age :",p2.age)  
  
  
#Q2  
class Student:  
 def \_\_init\_\_(self,name="John",age=19):  
 self.name=name  
 self.age=age  
  
 def getinfo(self):  
 print("Name : ",self.name)  
 print("Age : ",self.age)  
  
s1=Student()  
s2=Student("Spriha",18)  
s3=Student("ABCD",23)  
  
s1.getinfo()  
s2.getinfo()  
s3.getinfo()  
  
#Q3  
class account:  
 def \_\_init\_\_(self,bal):  
 self.bal=bal  
  
 def credit(self):  
 bal=self.bal  
 amount=int(input("Enter amount to be credited into account : "))  
 self.bal=bal+amount  
 print("Amount credited successfully.")  
 self.checkbalance()  
  
 def debit(self):  
 bal=self.bal  
 amount=int(input("Enter amount to be debited from account : "))  
 self.bal=bal-amount  
 print("Amount debited successfully.")  
 self.checkbalance()  
  
 def checkbalance(self):  
 print("Current balance : ",self.bal)  
  
a1=account(10000)  
a1.checkbalance()  
a1.credit()  
a1.debit()

**LAB 7**

#q1  
class person:  
 def \_\_init\_\_(self, name, age): # Correct method definition and parameters  
 self.name = name # Correct variable name  
 self.age = age  
  
 def display(self): # Correct method definition  
 print(f"Name: {self.name}, Age: {self.age}")  
  
  
class student(person):  
 def \_\_init\_\_(self, name, age, student\_id): # Correct method definition and parameters  
 super().\_\_init\_\_(name, age)  
 self.student\_id = student\_id  
  
 def show\_details(self):  
 self.display()  
 print(f"Student ID: {self.student\_id}")  
  
  
student1 = student("Alice", 20, "S12345")  
student1.show\_details()  
  
#q2  
class vehicle:  
 def intro(self): # Corrected method name  
 print("This is a vehicle")  
  
class car(vehicle):  
 def car\_info(self):  
 print("This is a car")  
  
class electriccar(car): # Corrected class name  
 def battery\_info(self):  
 print("This car has a battery")  
  
electric\_car = electriccar() # Corrected class name  
electric\_car.intro() # Corrected method call  
electric\_car.car\_info()  
electric\_car.battery\_info()  
  
#q3  
class teacher:  
 def description(self):  
 print("This is a teacher")  
  
class author:  
 def description(self):  
 print("This is an author") # Corrected spelling  
  
class tutorauthor(teacher, author):  
 def description(self):  
 super().description() # Calls the teacher's description  
 author.description(self) # Calls the author's description  
  
tutor\_author = tutorauthor() # Corrected variable name  
tutor\_author.description()  
  
#q4  
class animal:  
 def sound(self):  
 print("Animals make sound")  
  
class Dog(animal):  
 def sound(self):  
 print("Dog barks")  
  
class Cat(animal): # Corrected class name  
 def sound(self):  
 print("Cat meows")  
  
dog = Dog()  
cat = Cat()  
dog.sound()  
cat.sound() # Added parentheses

**LAB 8**

#q1  
class BubbleSort:  
 def \_\_init\_\_(self, array):  
 self.array = array  
  
 def sort(self):  
 n = len(self.array)  
 for i in range(n):  
 for j in range(0, n-i-1):  
 if self.array[j] > self.array[j+1]:  
 self.array[j], self.array[j+1] = self.array[j+1], self.array[j]  
  
array = [64, 34, 25, 12, 22, 11, 90]  
bs = BubbleSort(array)  
bs.sort()  
print(bs.array)  
  
#q2  
def insertion\_sort(array):  
 for i in range(1, len(array)):  
 key = array[i]  
 j = i - 1  
 while j >= 0 and key < array[j]:  
 array[j + 1] = array[j]  
 j -= 1  
 array[j + 1] = key  
 return array  
  
print(insertion\_sort([64, 34, 25, 12, 22, 11, 90]))  
  
#q3  
def selection\_sort(array):  
 for i in range(len(array)):  
 min\_idx = i  
 for j in range(i+1, len(array)):  
 if array[j] < array[min\_idx]:  
 min\_idx = j  
 array[i], array[min\_idx] = array[min\_idx], array[i]  
 return array  
  
print(selection\_sort([64, 34, 25, 12, 22, 11, 90]))  
  
#q4  
def merge\_sort(array):  
 if len(array) > 1:  
 mid = len(array) // 2  
 left\_half = array[:mid]  
 right\_half = array[mid:]  
  
 merge\_sort(left\_half)  
 merge\_sort(right\_half)  
  
 i = j = k = 0  
 while i < len(left\_half) and j < len(right\_half):  
 if left\_half[i] < right\_half[j]:  
 array[k] = left\_half[i]  
 i += 1  
 else:  
 array[k] = right\_half[j]  
 j += 1  
 k += 1  
  
 while i < len(left\_half):  
 array[k] = left\_half[i]  
 i += 1  
 k += 1  
  
 while j < len(right\_half):  
 array[k] = right\_half[j]  
 j += 1  
 k += 1  
 return array  
  
print(merge\_sort([64, 34, 25, 12, 22, 11, 90]))  
  
#q5  
def linear\_search(array, target):  
 for i in range(len(array)):  
 if array[i] == target:  
 return i  
 return -1  
  
print(linear\_search([64, 34, 25, 12, 22, 11, 90], 22))  
  
#q6  
def binary\_search(array, target):  
 left, right = 0, len(array) - 1  
 while left <= right:  
 mid = (left + right) // 2  
 if array[mid] == target:  
 return mid  
 elif array[mid] < target:  
 left = mid + 1  
 else:  
 right = mid - 1  
 return -1  
  
sorted\_array = [11, 12, 22, 25, 34, 64, 90]  
print(binary\_search(sorted\_array, 22))  
  
#q7  
class Node:  
 def \_\_init\_\_(self, key):  
 self.key = key  
 self.left = None  
 self.right = None  
  
class BinaryTree:  
 def \_\_init\_\_(self):  
 self.root = None  
  
 def insert(self, key):  
 if self.root is None:  
 self.root = Node(key)  
 else:  
 self.\_insert(self.root, key)  
  
 def \_insert(self, current, key):  
 if key < current.key:  
 if current.left is None:  
 current.left = Node(key)  
 else:  
 self.\_insert(current.left, key)  
 else:  
 if current.right is None:  
 current.right = Node(key)  
 else:  
 self.\_insert(current.right, key)  
  
 def inorder(self, node):  
 if node:  
 self.inorder(node.left)  
 print(node.key, end=" ")  
 self.inorder(node.right)  
  
bt = BinaryTree()  
elements = [50, 30, 70, 20, 40, 60, 80]  
for el in elements:  
 bt.insert(el)  
bt.inorder(bt.root)