Name: B L N Wajith Ali

USN: 22BTRAD009

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Module 1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 1]

F = eval(input())

x = eval(input())

k = F/x

print(" THe force constant of the spring is", k, "N/m")

# 1]

while True:

print("1] k=F/x")

print("2] F=k\*x")

print("3] x=F/k")

print("4] exit")

choice = int(input())

if choice == 1:

F = eval(input())

x = eval(input())

k=F/x

print(k)

elif choice == 2:

k = eval(input())

x = eval(input())

F=k\*x

print(F)

elif choice == 3:

F = eval(input())

k = eval(input())

x = F/k

print(x)

else:

exit()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 2]

import math

N = 100

t = 40

T = 0.4

m = 5

K = 4\*(math.pi\*\*2)\*(m/T\*\*2)

print(K)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 3]

import math

k = eval(input("enter k value: "))

m = eval(input("enter m value: "))

T = 2\*math.pi\*(math.sqrt(m/k))

f = 1/T

print(f)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# # 4]

import numpy as np

import matplotlib.pyplot as plt

A = int(input("enter A Value "))

f = int(input("enter f Value "))

phi = 0

sr = 100 # sampling rate

time = np.arange(0, 2, 1/sr) # 0.001 sec

x = A\* np.sin(2\*np.pi\*f\*time + phi)

plt.figure(figsize=(10,4))

plt.plot(time,x)

plt.title("Differential equation for Free Oscillations")

plt.xlabel("time(s)")

plt.ylabel("Displacement")

plt.show()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#5]

import math

import numpy as np

def num\_aperture(n1,n2,n0):

return (math.sqrt((n1\*n1)-(n2\*n2)))/n0

n1=eval(input("Enter the refractive indexof n1 = "))

n2=eval(input("Enter the refractive index of n2 = "))

n0=eval(input("Enter the refractive index of n0 = "))

num\_aper=num\_aperture(n1,n2,n0)

fract\_index=(n1-n2)/n1

accept\_angle=math.asin(num\_aper)

print("Numerical aperture of the fiber is ",num\_aper)

print("Accecptance angle is ",accept\_angle)

print("Fractional index is ",fract\_index)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#6]

import math

def num\_aperture(n1, n2, n0):

return (math.sqrt((n1 \* n1) - (n2 \* n2))) / n0

n1 = float(input("Enter the refractive index of n1: "))

n2 = float(input("Enter the refractive index of n2: "))

n0 = float(input("Enter the refractive index of n0: "))

num\_aper = num\_aperture(n1, n2, n0)

fract\_index = (n1 - n2) / n1

accept\_angle = math.asin(num\_aper)

print("Numerical aperture of the fiber is", num\_aper)

print("Acceptance angle is", accept\_angle)

print("Fractional index is", fract\_index)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Oscillations

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#1

# displacement-time graph for simple harmonic motion

import matplotlib.pyplot as plt

import numpy as np

#giving the values of Amplitude A, linear frequency f, and phase phi

A = int(input("enter the Amplitude A for the SHM "))

f = int(input("enter the linear frequency f for the SHM "))

phi = int(input("enter the phase phi for the SHM "))

#choosing the x-values(time interval) and sampling rate

time = np.arange(0, 0.2, 0.001)

y = A\* np.sin(2\*np.pi\*f\*time + phi)

#plot settings

plt.figure(figsize=(10,4))

plt.plot(time,y)

plt.title("displacement-time graph for simple harmonic motion")

plt.xlabel("time")

plt.ylabel("Displacement")

plt.show()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#2

# velocity-time-graph for simple harmonic motion

import matplotlib.pyplot as plt

import numpy as np

#giving the values of Amplitude A, linear frequency f, and phase phi

A = int(input("enter the Amplitude A for the SHM "))

f = int(input("enter the linear frequency f for the SHM "))

phi = int(input("enter the phase phi for the SHM "))

time = np.arange(0, 0.2, 0.001) # defining the x-axis values

y = A\* np.sin(2\*np.pi\*f\*time + phi) # displacement

v= 2\*np.pi\*f\*np.sqrt((A\*A)-(y\*y)) # velocity (y axis values)

#plot settings

plt.figure(figsize=(10,4))

plt.plot(time,v)

plt.title("velocity-time graph for simple harmonic motion")

plt.xlabel("time")

plt.ylabel("Velocity")

plt.show()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#3

# displacement-time graph for damped oscillations with b^2 > w^2 condition

import numpy as np

import matplotlib.pyplot as plt

#giving the values b, and omega w

A = 1

b = int(input("enter the damping coefficient b for the SHM "))

w = int(input("enter the angular frequency w for the SHM "))

# choosing the x-values(time interval) and sampling rate

time = np.arange(0, 2, 0.01) # 0.01 is the sampling rate

c = np.square(b)-np.square(w)

d = -b+np.sqrt(c)

e = -b-np.sqrt(c)

y = A\*np.exp(d\*time) + A\*np.exp(e\*time) # y axis values

#plot settings

plt.figure(figsize=(10,4))

plt.plot(time,y)

plt.title("displacement-time graph for damped oscillation")

plt.xlabel("time")

plt.ylabel("Displacement")

plt.show()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#4

# displacement-time graph for damped oscillations with b^2 = w^2 condition

import numpy as np

import matplotlib.pyplot as plt

#giving the values b, and omega w

A = 1

b = int(input("enter the damping coefficient b for the SHM "))

w = int(input("enter the angular frequency w for the SHM "))

# choosing the x-values(time interval) and sampling rate

time = np.arange(0, 2, 0.01) # 0.01 is the sampling rate

c = np.square(b)-np.square(w)

d = -b+np.sqrt(c)

e = -b-np.sqrt(c)

y = A\*np.exp(d\*time) + A\*np.exp(e\*time)

#plot settings

plt.figure(figsize=(10,4))

plt.plot(time,y)

plt.title("displacement-time graph for damped oscillation")

plt.xlabel("time")

plt.ylabel("Displacement")

plt.show()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#5

# displacement-time graph for damped oscillations with b^2 < w^2

import cmath

import numpy as np

import matplotlib.pyplot as plt

A = 1

b = float(input("enter the damping coefficient b for the SHM "))

w = int(input("enter the angular frequency w for the SHM "))

time = np.arange(0, 2, 0.01) # 0.01 is the sampling rate

c = np.square(b)-np.square(w)

d = -b+cmath.sqrt(c)

e = -b-cmath.sqrt(c)

x = A\*np.exp(d\*time) + A\*np.exp(e\*time)

plt.figure(figsize=(5,4))

plt.plot(time,x)

plt.title("displacement-time graph for under damped oscillations")

plt.xlabel("time(s)")

plt.ylabel("Amplitude")

plt.show()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Module 2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#1

# The ground state energy of an electron in a one-dimensional box is 5.6 meV.

# What will be the ground state energy if the width of the well is doubled?

import numpy as np

#giving n, L and m values

E1=float(input("enter the ground state energy value in meV: "))

# defining the function

Enew = (1/4)\*E1

print("The new ground state energy value when the width of the well doubled is= ", Enew,"J")

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#2

# Calculate the first 3 permitted energy values for an electron in a box of width 4 𝐴 ̇.

# importing the required modules

import numpy as np

#defining the constants h

h = 6.626e-34

m = 9.11e-31

k = 10e19

#giving n, L and m values

L=int(input("enter the width of the well 'L' in angstrom unit "))

# defining the function

E0 = ((np.square(1)\*np.square(h))/(8\*m\*np.square(L)))\*k

E1 = ((np.square(4)\*np.square(h))/(8\*m\*np.square(L)))\*k

E2 = ((np.square(9)\*np.square(h))/(8\*m\*np.square(L)))\*k

print("The zero point energy is", E0, J)

print("The first excited state energy is", E1, J)

print("The second excited state energy is", E2, J)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#3

# The velocity of an electron was measured to be 5"x" 10^5 m/s with an uncertainty of 1%.

# What is the uncertainty involved in the measurement of position.

# importing the required modules

import numpy as np

#defining the constants h

h = 6.626e-34

m = 9.11e-31

k = 10e19

V= 5 \* 10e19

deltaV = (V \* 1)/100

deltaP = m \* deltaV

deltaX = h / 4\*np.pi\*deltaP

print("The uncertainty involved in the measurement of position is", deltaX, m)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#4

# importing the required modules

import matplotlib.pyplot as plt

import numpy as np

#title of the figure

#giving n, L values

n=int(input("enter the n value "))

L=int(input("enter the width of the well L in angstrom unit "))

# setting the x - coordinates

x = np.arange(0, 1, 0.01)

# setting the y - coordinates for both the plots

psi = np.sqrt(2/L)\*np.sin(n\*np.pi\*x/L)

ps = np.square(psi)

# setting the plot dimensions

plt.figure(figsize=(10,3))

# plotting the left graph

plt.subplot(1, 2, 1) # row 1, col 2 index 1

plt.plot(x, psi)

plt.title("Wavefunction plot for n="+str(n))

plt.xlabel('L')

plt.ylabel('Ψ')

# plotting the Right graph

plt.subplot(1, 2, 2) # index 2

plt.plot(x, ps)

plt.title("corresponding probability density plot for n="+str(n))

plt.xlabel('L')

plt.ylabel('Ψ \* Ψ')

plt.show()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Module 3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#1

def photons(p,h,c,x,t):

return (p\*x\*t)/(h\*c)

x=eval(input("Enter the wavelength of the radiation = "))

p=eval(input("Enter the avg power = "))

t=eval(input("Enter the time = "))

h=6.63e-34

# h=6.625e-34

c=3e8

n=photons(p,h,c,x,t)

print("No of photons =",n)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#2

import math

def ratio(h,c,x,k,T):

return math.exp(-(h\*c)/(x\*k\*T))

x=eval(input("Enter the wavelength = "))

T=eval(input("Enter the temperature = "))

h=6.63e-34

k=1.38e-23

c=3e8

m=ratio(h,c,x,k,T)

print("The ratio population is ", m)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#3

def wavelength(n,h,c,P,t):

return (n\*h\*c)/(P\*t)

n=eval(input("Enter the no of photons = "))

P=eval(input("Enter the Power = "))

t=eval(input("Enter the time taken = "))

h=6.63e-34

c=3e8

print("The wavelength of the laser beam is", wavelength(n,h,c,P,t))

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#4

n=eval(input("Enter the number of photons = "))

lam=eval(input("Enter the wavelength = "))

t=eval(input("The time taken = "))

pi=eval(input("Enter the input power = "))

h=6.63e-34

c=3e8

po=(n\*h\*c)/(lam\*t)

efficiency=(po/pi)\*100

print("The percent power consumed into coherent light is ", efficiency)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#5

import matplotlib.pyplot as plt

import numpy as np

import math

#giving the values of temperature, kb

kb=1.38e-23

T=eval(input("Enter the temperature: "))

#choosing the x-vakyes(delta E value) and sampling rate

E=np.arange(0,0.01,0.001)

y=np.exp(-(E)/(kb\*T))

#plot settings

plt.figure(figsize=(10,4))

plt.plot(E,y)

plt.title("variation of population ratio with E")

plt.xlabel("E")

plt.ylabel("Population density ratio")

plt.show()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Module 4

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#1]  
# A He-Ne laser is emitting a beam with an average power of 4.5 mW. Find the number of photons emitted per second by the laser.

# The wavelength of the emitted radiation is 6328 A⁰.

import numpy as np

t = float(input("enter the time t: ",))

p = float(input("enter the power of the laser beam p: "))

λ = float(input("enter the wavelength λ: "))

h = 6.63e-34

c = 3e8

N = (p\*t\*λ)/(h\*c)

print("the number of photons emitted per second by the laser is: ", N)

#2]

# Find the ratio of population of two energy levels in a laser medium if the transition between them produces light of

# wavelength 694.3 nm. Assume the ambient temperature to be 27⁰ C.

import numpy as np

import math

T = float(input("enter the tempearature T : "))

λ = float(input("enter λ Value : "))

k = 1.38e-23

h = 6.63e-34

c = 3e8

Q = (h\*c)/(λ\*k\*T)

N = math.exp(-Q)

print("The ratio of population of two energy levels is : ", N)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#3

# A laser beam with power per pulse is 1.0 mW lasts 10 ns.

# If the number of photons emitted per pulse is 3.941 × 10⁷, Calculate the wavelength of laser.

import numpy as np

t = float(input("enter the time t : "))

P = float(input("enter the power of the laser pulse p : "))

N = float(input("enter the number of photons N : "))

h = 6.63e-34

c = 3e8

λ = (N\*h\*c)/(P\*t)

print("the wavelength of the laser is : ", λ, "m")

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#4

# A laser operating at 632.8 nm emits 3.182×10¹⁶ photons per second.

# Calculate the output power If the laser input power is 100 watt. Also find the percentage power converted into coherent light energy

import numpy as np

N = float(input("enter the number of photons emitted per second : "))

λ = float(input("enter the wavelength : "))

IP = float(input("enter the input power : "))

h = 6.63e-34

c = 3e8

#Energy of one photon

E = (h\*c)/λ

#Output Power(A)

OP = N\*E

print("the output power is : ", OP)

# efficiency

e = (OP/IP)\*100

print("the percentage power converted into coherent light energy is: ", e)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#5

import matplotlib.pyplot as plt

import numpy as np

import math

#giving the values of temperature, kb

kb = 1.38e-23

T = float(input("enter the temperature : "))

#choosing the x-values(delta E value) and sampling rate

E = np.arange(0, 0.01, 0.001)

y = np.exp(-(E)/(kb\*T))

#plot settings

plt.figure(figsize=(10,4))

plt.plot(E,y)

plt.title("variation of population ratio with E")

plt.xlabel("E")

plt.ylabel("Population density ratio")

plt.show()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Module 5

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#1]

import sympy.logic.boolalg

while True:

print("1] For AND gate")

print("2] For OR gate")

print("3] For NOT gate")

print("4] For NAND gate")

print("5] For NOR gate")

print("6] For EX-OR gate")

print("7] For EX-NOR gate")

print("8] Exit")

choice = int(input("Enter your choice: "))

if choice >= 1 and choice <= 7:

inputs = input("Enter your inputs (Only '0' and '1' allowed): ")

inputs = inputs.replace(" ", "")

print("True = 1 and False = 0")

if all(bit in ["0", "1"] for bit in inputs):

if choice == 1:

result = int(inputs[0])

for bit in inputs[1:]:

result = sympy.logic.boolalg.And(result, int(bit))

print("AND gate output:", result)

elif choice == 2:

result = int(inputs[0])

for bit in inputs[1:]:

result = sympy.logic.boolalg.Or(result, int(bit))

print("OR gate output:", result)

elif choice == 3:

if inputs == "0":

result = 1

elif inputs == "1":

result = 0

else:

print("Invalid input for NOT gate.")

continue

print("NOT gate output:", result)

elif choice == 4:

result = int(inputs[0])

for bit in inputs[1:]:

result = sympy.logic.boolalg.Nand(result, int(bit))

print("NAND gate output:", result)

elif choice == 5:

result = int(inputs[0])

for bit in inputs[1:]:

result = sympy.logic.boolalg.Nor(result, int(bit))

print("NOR gate output:", result)

elif choice == 6:

result = int(inputs[0])

for bit in inputs[1:]:

result = sympy.logic.boolalg.Xor(result, int(bit))

print("EX-OR gate output:", result)

elif choice == 7:

result = int(inputs[0])

for bit in inputs[1:]:

result = sympy.logic.boolalg.Xnor(result, int(bit))

print("EX-NOR gate output:", result)

else:

print("Invalid input. Only '0' and '1' are allowed.")

elif choice == 8:

exit()

else:

print("Invalid input. Please select a number from 1 to 7.")

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_