MUHAMMAD UMAR KHAN 10619
NC LAB#04

Question No: 01

```
In [7]: f1=lambda x:x**2-3

In [8]: f2=lambda x:x**3-x-1

In [9]: f3=lambda x:x**3+1

In [10]: f4=lambda x:x**3+4*(x**2)-10

In [11]: def secant(fun,x0,x1,tol=0.0001,niter=50):
             for it in range (niter):
                 x2=x0-(x1-x0)/(fun(x1)-fun(x0))*fun(x0)
                 if abs(x2-x1)<tol:break
                 else:
                     x0=x1
                     x1=x2
             else:
                 print("Maximum iterations finished and secant method is diverging")
             return x2,it
```

```
In [15]: x0=float(input("enter x0: "))
         x1=float(input("enter x1: "))
         tol=0.001
         niter=20

         print("No# Function      interval  tol  no.of iterations   root")

         ans,iterations=secant(f1,x0,x1,tol,niter)
         print("1  x**2-3           1,2  ",tol,"    ",iterations,"    ",ans)

         ans,iterations=secant(f2,x0,x1,tol,niter)
         print("2  x**3-x-1         1,2  ",tol,"    ",iterations,"   ",ans)

         ans,iterations=secant(f3,x0,x1,tol,niter)
         print("3  x**3+1           1,2  ",tol,"    ",iterations,"    ",ans)

         ans,iterations=secant(f4,x0,x1,tol,niter)
         print("4  x**3+4*(x**2)-10  1,2  ",tol,"    ",iterations,"     ",ans)
```

```
enter x0: 15
enter x1: 20
No# Function      interval  tol  no.of iterations   root
1  x**2-3           1,2   0.001      8      1.7320509001766573
2  x**3-x-1         1,2   0.001      13     1.3247180025857659
3  x**3+1           1,2   0.001      18      -0.9999999488365837
4  x**3+4*(x**2)-10  1,2   0.001      11       1.3652300411746965
```

Ser. No.	Function	Strating Interval	Tolerance	No. of Iterations	Root
1	x^2-3=0	1,2	0.001	8	1.7320509001766573
2	x^3–x–1=0	1,2	0.001	13	1.3247180025857659

3	**x^3+1=0**	**1,2**	**0.001**	**18**	**-0.9999999488365837**
4	**x^3+4*x^2-10=0**	**1,2**	**0.001**	**11**	**1.3652300411746965**

Question No: 02

```
import math
f = lambda x:x**3-0.165*x**2+3.993*(math.e**-4)

def bisection(a,b,t):
    niter=0
    while (abs(a-b)>=t):
        mid=(a+b)/2.0
        prod=func(a)*func(mid)
        if prod>t:
            a=mid
        else:
            if prod<t:
                b=mid
        niter+=1
    return mid,niter

def RF(fun,a,b,niter,tol,verbose=False):
    if fun(a) * fun(b)>0:
        c=None
        msg="Invalid points"
    else:
        for i in range(niter):
            fa=fun(a)
            fb=fun(b)
            c=(a*fb-b*fa)/(fb-fa)
            fc=fun(c)
            if verbose:
                print("Root is {} after {} iterations".format(c,i))
            msg="Maximum iterations completed"
            if abs(fc)<tol:
                msg="Root found!!"
                break
            elif fa*fc<0:
                b=c
            elif fb*fc<0:
                a=c
    return c,i

def secant(fun,x0,x1,tol,niter=50):
    for i in range(niter):
        x2=x0-(x1-x0)/(fun(x1)-fun(x0))*fun(x0)
        if abs(x2-x1)<tol: break
        else:
            x0=x1
            x1=x2
    else:
        print("Maximum iteratons finished and secant method is diverging")
    return x2,i
```

```
ab,ib=bisection(-1,2,0.01)
arf,irf=RF(f,-1,2,50,0.01)
ans,its=secant(f,-1,2,0.01)
```

MUHAMMAD UMAR KHAN 10619
NC LAB#04

```
if ib < irf and ib < its:
    print('Bisection Method Is Efficient')
    print('Root: ',ab,' Iterations: ',ib)
elif irf < ib and irf < its:
    print('Regula Farsi Method Is Efficient')
    print('Root: ',arf,' Iterations: ',irf)
elif its < irf and its < ib:
    print('Secant Method Is Efficient')
    print('Root: ',ans,' Iterations: ',its)
```

```
Secant Method Is Efficient
Root:  -0.3698855790669576  Iterations:  5
```

```
print('Bisection Method:')
print('Root: ',ab,' Iterations: ',ib)
print('Regula Farsi Method: ')
print('Root: ',arf,' Iterations: ',irf)
print('Secant Method: ')
print('Root: ',ans,' Iterations: ',its)
```

```
Bisection Method:
Root:  -0.501953125  Iterations:  9
Regula Farsi Method:
Root:  -0.38674786101532665  Iterations:  11
Secant Method:
Root:  -0.3698855790669576  Iterations:  5
```