```python
#Q1
#Bisection Method
def f(x):
    return 2*x**2 + 6*x + 4

def bisection_method(f, a, b, tol):
    n = 0
    while abs(b - a) > tol:
        g = (b + a)/2
        n += 1
        if f(a) * f(g) < 0:
            b=g
        else:
            a=g
    return (g, n)

# bisection_method(f, -3, -1.5, 1e-5)
print(bisection_method(f, -3, -1.5, 1e-5))
```

```
(-1.9999980926513672, 18)
```

```python
#Fixed-Point Method
import numpy as np
def g(x):
    return -2/(x+3)

def fixed_point(g, p, tol):
    n = 0
    p1 = p
    p2 = g(p1)
    while abs(p2 - p1) > tol:
        n += 1
        p1 = p2
        p2 = g(p1)
    return (p2, n)

# fixed_point(g, 2, 1e-5)
print(fixed_point(g, 2, 1e-5))
```

```
(-0.9999942779213596, 16)
```

```python
#Newton's Method
def f(x):
    return 2*x**2 + 6*x + 4

def df(x):
    return 4*x + 6

def newton(f, df, p, tol):
    n = 0
    x1 = p
    x2 = x1 - f(x1)/df(x1)
    while abs(x2 - x1) > tol:
        n += 1
        x1 = x2
        x2 = x1 - f(x1)/df(x1)
    return (x2, n)

# newton(f, df, 2, 1e-5)
print(newton(f, df, 2, 1e-5))
```

```
(-0.9999999999999996, 6)
```

```python
#Q2
#Fixed-Point Method
import numpy as np
def g(x):
    return (np.exp(x)+2)/5

def fixed_point(g, p, tol):
    p1 = p
    p2 = g(p1)
    while abs(p2 - p1) > tol:
        p1 = p2
        p2 = g(p1)
    return p2

fixed_point(g, 2, 1e-5)
```

```
0.8842242800356657
```

```python
#Q3
#Bisection Method
def f(x):
    return np.exp(x) - 5*x + 2

def bisection_method(f, a, b, tol):
    while abs(b - a) > tol:
        g = (b + a)/2
        if f(a) * f(g) < 0:
            b=g
        else:
            a=g
    return g

bisection_method(f, 2, 3, 1e-5)
```

```
2.1937484741210938
```

```python
#Newton's Method
def f(x):
    return np.exp(x) - 5*x + 2

def df(x):
    return np.exp(x) - 5

def newton(f, df, p, tol):
    x1 = p
    x2 = x1 - f(x1)/df(x1)
    while abs(x2 - x1) > tol:
        x1 = x2
        x2 = x1 - f(x1)/df(x1)
    return x2

newton(f, df, 1, 1e-5)
```

```
0.8842181389559761
```

```python
#Q4
#Bisection Method
def f(x):
    return x**3 + 2*x**2 + 10*x - 20

def bisection_method(f, a, b, tol):
    while abs(b - a) > tol:
        g = (b + a)/2
        if f(a) * f(g) < 0:
            b=g
        else:
            a=g
    return g

bisection_method(f, 0, 2, 1e-5)
```

```
1.3688125610351562
```