Section 2.3

Question 6

```
def newton( x, f, df, E ):
    h = f(x) / df(x)
    n=0
    while abs(h) >= E:
        h = f(x)/df(x)
        x = x - h
        n+=1

    print("The value of the root is : ",
                             "%.10f"% x)
    print("The number of iterations is : ", "%d"%n)


def f1(x):
  return math.log(x-1) + math.cos(x-1)
def df1(x):
  return 1/(x-1) - math.sin(x-1)
newton(1.3,f1,df1, 0.00001)
```

```
The value of the root is :  1.3977484760
The number of iterations is :  4
```

```
def f2(x):
  return (x-2)*(x-2) - math.log(x)
def df2(x):
  return 2*(x-2) - 1/x
newton(1,f2,df2,0.00001)
newton(math.exp(1),f2,df2,0.00001)
```

```
The value of the root is :  1.4123911720
The number of iterations is :  4
The value of the root is :  3.0571035500
The number of iterations is :  5
```

Question 8

```
def secant(f, p1, p2, E):
    n = 0;
    pm = 0;
    p0 = 0;
    c = 0;
    if (f(p1) * f(p2) < 0):
        while True:
            p0 = ((p1 * f(p2) - p2 * f(p1)) /
                            (f(p2) - f(p1)));
            c = f(p1) * f(p0);
            p1 = p2;
            p2 = p0;
            n += 1;
            if (c == 0):
                break;
            pm = ((p1 * f(p2) - p2 * f(p1)) /
                            (f(p2) - f(p1)));

            if(abs(pm - p0) < E):
                break;

        print("Root of the given equation =",
                               round(p0, 6));
        print("Number of iterations = ", n);

    else:
        print("Can not find a root in ",
                   "the given inteval");



def f1(x):
  return math.log(x-1) + math.cos(x-1)
secant(f1,1.3,2,0.00001)
```

```
Root of the given equation = 1.397749
Number of iterations =  7
```

```
def f2(x):
  return (x-2)*(x-2) - math.log(x)
secant(f2, 1, 2, 0.00001)
secant(f2, math.exp(1), 4, 0.00001)
```

```
Root of the given equation = 1.412391
Number of iterations =  6
Root of the given equation = 3.057103
Number of iterations =  5
```

Question 15

```
def f3(x):
  return 4*x*x - math.exp(x) - math.exp(-x)
def df3(x):
  return 8*x - math.exp(x) + math.exp(-x)
#(a)
newton(-10, f3, df3, 0.00001)
```

```
The value of the root is :  -4.3062452735
The number of iterations is :  11
```

```
#(b)
newton(-5, f3, df3, 0.00001)
```

```
The value of the root is :  -4.3062452735
The number of iterations is :  5
```

```
#(c)
newton(-3, f3, df3, 0.00001)
```

```
The value of the root is :  0.8244985853
The number of iterations is :  5
```

```
#(d)
newton(-1, f3, df3, 0.00001)
```

```
The value of the root is :  -0.8244985853
The number of iterations is :  4
```

```
#(e)
newton(0, f3, df3, 0.00001)
# Fail since df3(0) = 0 Newton's method cannot implement
```

```
#(f)
newton(1, f3, df3, 0.00001)
```

```
The value of the root is :  0.8244985853
The number of iterations is :  4
```

```
#(g)
newton(3, f3, df3, 0.00001)
```

```
The value of the root is :  -0.8244985853
The number of iterations is :  5
```

```
#(h)
newton(5, f3, df3, 0.00001)
```

```
The value of the root is :  4.3062452735
The number of iterations is :  5
```

```
#(i)
newton(10, f3, df3, 0.00001)
```

```
The value of the root is :  4.3062452735
The number of iterations is :  11
```