```
Alexander Winkles Computer Project #2
09/12/2016

--------------------------------------------------------------------------------
Problem 1:
--------------------------------------------------------------------------------

Here is a copy of my code:

type newton.m

function newton(f,df,x0,T,S,q)
%newton(f,df,x0,T,S,q)
%
%This is a Newton's method algorithm developed by Alexander Winkles used to
% find roots in functions.
%
%f  : the function being studied.
%df : the derivative of the function being studied.
%x0 : the initial point being used.
%T  : the tolerance of the final answer.
%S  : the number of iterations.
%q  : prints the steps (1 for yes, 0 for no).

format long
if q == 1
    fprintf('\nStep \t\t Result\n---\t\t --------\n');
end;
if feval(f,x0) == 0 || abs(feval(f,x0)) < T
    fprintf('\n Your inital guess %d was close enough to zero!',x0);
else
    i = 0;
    while i <= S
        x1 = x0 - feval(f,x0)/feval(df,x0);
        if q == 1
            fprintf('%d \t\t %f\n',i,x1);
        end;
        if feval(f,x1) == 0 || abs(feval(f,x1)) < T
            fprintf('\nThe solution is %d. The computation was a success
                after %d iterations!\n\n',x1,i);
            break;
        end;
        i = i+1;
        x0 = x1;
    end;
    if i == S+1
        fprintf('\nMethod failed after %d iterations.\n\n',S)
    end;
end;

Inputing the function and it's derivative:

f =@(x) tan(x);
```

```
df =@(x) (sec(x))^2;

Computing the results:

newton(f,df,4.5,1e-5,100,1)

Step                  Result
---                  --------
0                   4.293941
1                   3.922657
2                   3.422676
3                   3.156166
4                   3.141595

The solution is 3.141595e+00. The computation was a success after 4 iterations!

newton(f,df,7.7,1e-5,100,1)

Step                  Result
---                  --------
0                   7.548441
1                   7.261564
2                   6.798345
3                   6.369614
4                   6.283615
5                   6.283185

The solution is 6.283185e+00. The computation was a success after 5 iterations!

--------------------------------------------------------------------------------
Problem 4:
--------------------------------------------------------------------------------

Inputing the function and its derivative:

f =@(x) x.^3-5*x.^2+3*x-7;

df =@(x) 3*x.^2-10*x+3;

Computing the results:

newton(f,df,5,1e-5,10,1)

Step                  Result
---                  --------
0                   4.714286
1                   4.679089
2                   4.678574

The solution is 4.678574e+00. The computation was a success after 2 iterations!

This solution was found before 10 steps, so we use a new program to compute it to 10 steps:

type newtonstep.m
```

```
function newtonstep(f,df,x0,S)
%newtonstep(f,df,x0,T,S)
%
%This is a program derived from Newton's method algorithm developed by Alexander Winkles used to
% find roots in functions for a certain number of steps.
%
%f  : the function being studied.
%df : the derivative of the function being studied.
%x0 : the initial point being used.
%S  : the number of steps required.

format long

i = 0;
fprintf('\nStep \t\t Result\n---\t\t --------\n');
while i <= S
    x1 = x0 - feval(f,x0)/feval(df,x0);
    fprintf('%d \t\t %f\n',i,x1);
    i = i+1;
    x0 = x1;
end;
fprintf('\nAfter %d steps, the solution is %d.\n\n',i-1,x1)
```

Which gave these results:

```
newtonstep(f,df,5,10)

Step                  Result
---                  --------
0                   4.714286
1                   4.679089
2                   4.678574
3                   4.678574
4                   4.678574
5                   4.678574
6                   4.678574
7                   4.678574
8                   4.678574
9                   4.678574
10                   4.678574

After 10 steps, the solution is 4.678574e+00.
```

Problem 9:

Here is my code:

```
type steffensen.m

function steffensen(f,x0,T,S,q)
%steffensen(f,x0,T,S,q)
```

```
%
%This is a Steffensen's method algorithm designed by Alexander Winkles to
% find roots of functions.
%
%f  : the function being evaluated.
%x0 : the initial value being used.
%T  : the tolerance of the final answer.
%S  : the number of iterations.
%q  : prints the steps (1 for yes, 0 for no)
%
%*Current issues: does not necessarily converge to the closest zero from the
% starting point.

format long
if q == 1
    fprintf('\nStep \t\t Result\n---\t\t --------\n');
end;
if abs(feval(f,x0)) < T
    fprintf('\n Your inital guess %d was close enough to zero!',x0);
else
    i = 0;
    while i<= S
        xi = x0 + feval(f,x0);
        x1 = x0 - feval(f,x0)/((feval(f,xi)-feval(f,x0))/feval(f,x0));
        if q == 1
            fprintf('%d \t\t %f\n',i,x1)
        end;
        if abs(feval(f,x1)) < T
            fprintf('\nThe solution is %f. The computation was successful
                after %d iterations!\n\n',x1,i)
            break;
        end;
        i = i+1;
        x0 = x1;
    end;
    if i == S+1
        fprintf('\nThe computation failed after %d iterations.\n\n',S)
    end;
end;
```

Here is my function:

```
f =@(x) tan(x);
```

Here are my results for testing:

```
steffensen(f,4.5,1e-5,100,1)

Step                  Result
---                  --------
0                   8.859416
1                   9.067304
2                   9.333973
3                   9.423281
```

```
4                  9.424778

The solution is 9.424778. The computation was successful after 4 iterations!

steffensen(f,7.7,1e-5,100,1)

Step                  Result
---                  --------
0                   7.928445
1                   -4.507402
2                   -9.211639
3                   -9.405459
4                   -9.424764
5                   -9.424778

The solution is -9.424778. The computation was successful after 5 iterations!

f=@(x) tan(x) - x;
steffensen(f,7.7,1e-5,100,1)

Step                  Result
---                  --------
0                   8.014521
1                   -1.820315
2                   1.983972
3                   -0.375582
4                   -0.261593
5                   -0.178073
6                   -0.119858
7                   -0.080252
8                   -0.053605
9                   -0.035767
10                   -0.023854

The solution is -0.023854. The computation was successful after 10 iterations!

steffensen(f,4.5,1e-5,100,1)

Step                  Result
---                  --------
0                   4.497787
1                   4.495335
2                   4.493781
3                   4.493423
4                   4.493409

The solution is 4.493409. The computation was successful after 4 iterations!

It appears that this method find zeros, but they are not always the closest
    ones to the starting value.

--------------------------------------------------------------------------------
Problem 13:
--------------------------------------------------------------------------------
```

Here is my code:

type newtonsystem.m

```
function newtonsystem(F,J,x0,y0,T,S,q)
%newtonsystem(F,J,x0,y1,T,S,q)
%
%This is a Newton's method algorithm developed by Alexander Winkles used to
% find roots in systems of nonlinear equations.
%
%F  : the function matrix being studied.
%J  : the Jacobian of the function matrix being studied.
%x0 : the initial vector being used.
%T  : the tolerance vector of the final answer.
%S  : the number of iterations.
%q  : prints the steps (1 for yes, 0 for no).

format long g

if q == 1
    fprintf('\nStep \t\t x \t\t\t y\n---\t\t -------- \t\t --------\n');
end;
if abs(feval(F,x0,y0)) < T
    fprintf('\nYour inital guess (%d %d) was close enough to zero!',x0,y0);
else
    i = 0;
    while i <= S
        H = -inv(J(x0,y0))*F(x0,y0);
        x1 = x0 + H(1);
        y1 = y0 + H(2);
        if q == 1
            fprintf('%d \t\t %f \t\t %f\n',i,x1,y1);
        end;
        if abs(feval(F,x1,y1)) < T
            fprintf('\nThe solution is (%f, %f). The computation was a success
                after %d iterations!\n\n',x1,y1,i);
            break;
        end;
        i = i + 1;
        x0 = x1;
        y0 = y1;
    end;
    if i == S+1
        fprintf('\nMethod failed after %d iterations.\n\n',S)
    end;
end;
```

Here is my function matrix and its Jacobian:

```
F =@(x,y) [1+x.^2-y.^2 + exp(x)*cos(y); 2*x*y+exp(x)*sin(y);];

J =@(x,y) [1+2*x+exp(x)*cos(y), 1-2*y-exp(x)*sin(y); 2*y+exp(x)*sin(y), 2*x+exp(x)*cos(y);];
```

```
Here are my results:

newtonsystem(F,J,-1,4,[1e-5;1e-5],100,1)

Step                x                       y
---                 --------                --------
0                  -0.565863               1.801265
1                  -0.409870               1.137365
2                  -0.273568               1.252288
3                  -0.296764               1.130502
4                  -0.290827               1.198308
5                  -0.294538               1.159211
6                  -0.292414               1.180442
7                  -0.293586               1.168392
8                  -0.292928               1.175071
9                  -0.293294               1.171320
10                  -0.293089               1.173412
11                  -0.293204               1.172240
12                  -0.293140               1.172895
13                  -0.293176               1.172529
14                  -0.293156               1.172733
15                  -0.293167               1.172619
16                  -0.293160               1.172683
17                  -0.293164               1.172647
18                  -0.293162               1.172667
19                  -0.293163               1.172656
20                  -0.293162               1.172662

The solution is (-0.293162, 1.172662). The computation was a success after 20 iterations!

Notice if we let z = x + y*i, f(z) = 1 + z^2 + exp(z) becomes
    1 + x^2 + 2*x*y*i - y^2 + exp(x)*(cos(y) + i*sin(y)).
Thus, problem 13 is problem 11, with each function representing either the
    real or the imaginary part of f(z).

diary off
```