Code

Newton Function

```
function [root] = Newton(x0,max_Step)
%% NEWTON Determines root using Newton's method
%% Requires initial value x0 and max_step; returns root.
 % Print header (Each col is 4 space wide)
 fprintf('%s  %s \n', 'step', 'x');

 % Set x = initial x, i.e. x0
 x =x0 ;

 % Run loop for 1 to max_step
 for i=1:max_Step
     % Evaluate fx = x^2 - 3x + 2
     fx = x^2 - 3*x + 2;
     % Evaluate dfx
     dfx = 2*x - 3;

     % Find x_new from x, fx, dfx using the
     x_new = x - fx/dfx;

     % Update x with x_new
     x =x_new ;

     % Print step number (i) and x.
     % Col1: integer, 4 spaces wide
     % Col2: float, 8 spaces wide with 4 decimal places

     fprintf('%d \t%4.4f\n', i,x);
 end

 root = x;

 fprintf('\n\n');
end
```

HW7 file code

```
clc; clear;
max_step = 5;
x0 = 1; % set starting value to 1
disp(['x0=',num2str(x0),' and max_step=',num2str(max_step)])
root = Newton(x0,max_step);

% Try again, this time with 2
x0 = 3;
disp(['x0=',num2str(x0),' and max_step=',num2str(max_step)])
root = Newton(x0,max_step);
% when we use x0=2 we get the root is 2 in all step ecause exect root is 2

% And try again, this time with x0 = 300. Did you get the
% result in 5 steps? If not, then what different
% can you do to get either 1 or 2 as the estimated root?
```

```
x0 = 300;
% max_step = ??;
disp(['x0=',num2str(x0),' and max_step=',num2str(max_step)])
root=Newton(x0,max_step);

% no by using 5 step we not get the result
% initial value is large and it reduce in every step almost half


x0 = 300;
max_step = 15;
disp(['x0=',num2str(x0),' and max_step=',num2str(max_step)])
root=Newton(x0,max_step);
```

Output

```
x0=1 and max_step=5
step  x
1   1.0000
2   1.0000
3   1.0000
4   1.0000
5   1.0000


x0=3 and max_step=5
step  x
1   2.3333
2   2.0667
3   2.0039
4   2.0000
5   2.0000


x0=300 and max_step=5
step  x
1   150.7504
2   76.1260
3   38.8147
4   20.1607
5   10.8370
```

```
x0=300 and max_step=15
step  x
1   150.7504
2   76.1260
3   38.8147
4   20.1607
5   10.8370
6   6.1819
7   3.8677
8   2.7366
9   2.2194
10  2.0335
11  2.0010
12  2.0000
13  2.0000
14  2.0000
15  2.0000
```