str = input('Input non-linear function: ','s');  
func = str2func(['@(x)' str]);  
start1 = input('Input 1st starting point: ');  
start2 = input('Input 2nd starting point: ');  
maxRelError = input('Input maximum allowed relative approximate error(in %): ');  
convgCriteria = input('Input convergence criterion for function value: ');  
maxIter = input('Input maximum number of iterations: ');  
figure  
fplot(func,[(a-10) (b+10)]);  
grid on;  
title('func(x)vs x')  
print -djpg BisectionFunction.jpg  
i = 0;  
if func(start1)\*f(start2)>0  
disp('starting points are incorrectly chosen');  
else  
oldError = (start1 + start2)/2;  
i = i + 1;  
err = 100;  
while ((err > maxRelError) && (i < maxIter))  
if func(start2)\*func(oldError)<0  
start1 = oldError;  
else  
start2 = oldError;  
end  
newError = (start1 + start2)/2;  
err = abs((newError - oldError)/newError)\*100;  
e(1,i)= err;  
oldError = newError;  
if(abs(func(newError))<convgCriteria)  
break;  
end  
i = i + 1;  
end  
end  
disp (oldError);  
if i >= maxIter  
disp('Maximum Iteration number attained.');  
elseif err <= maxRelError  
disp('Convergence for maximum relative approximate error reached.');  
else  
disp('Convergence criteria for function value reached.');  
end  
figure  
subplot(2,1,2)  
plot (1:itr-1,e(1:itr-1)) % error plot  
grid on;  
title('BisectionError vs Iteration')  
print -djpg BisectionError.jpg

str = input('Input non-linear function: ','s');  
func = str2func(['@(x)' str]);  
low = input('Input smaller starting point: ');  
high = input('Input bigger starting point: ');  
maxRelError = input('Input maximum allowed relative approximate error(in %): ');  
convgCriteria = input('Input convergence criterion for function value: ');  
maxIter = input('Input maximum number of iterations: ');  
figure  
fplot(func,[(low - 10) (high + 10)]);  
grid on;  
title('func(x) vs x')  
print -djpg FalsePositionFunc.jpg  
i = 0;  
m = low - func(low) \* ((high - low)/(func(high) - func(low)));  
i = i + 1;  
error = 100;  
while((error > maxRelError) && (i < maxIter))  
if func(high) \* func(m)<0  
low = m;  
else  
high = m;  
end  
n = low - func(low) \* ((high - low)/(func(high) - func(low)));  
err = abs((n - m)/n) \* 100;  
e(1, i)= error;  
m = n;  
if(abs(func(n))<convgCriteria)  
break;  
end  
i = i +1;  
end  
disp (m);  
if i >= maxIter  
disp('Maximum Iteration number attained.');  
elseif error <= maxRelError  
disp('Convergence for maximum relative approximate error reached.');  
else  
disp('Convergence criteria for function value reached.');  
end  
figure  
subplot(2,1,2)  
plot (1:itr-1,e(1:itr-1))  
grid on;  
title('FalsePositionError vs Iteration')  
print -djpg FalsePositionError.jpg

str = input('Input non-linear function: ','s');  
func = str2func(['@(x)' str2]);  
start = input('Input a starting point: ');  
maxRelError = input('Input maximum allowed relative approximate error(in %): ');  
convgCriteria = input('Input convergence criterion for function value: ');  
maxIter = input('Input maximum number of iterations: ');  
figure  
fplot(func,[(start - 10) (start + 10)]);  
grid on;  
title('func(x)vs x')  
print -djpg FixedPointFunc.jpg  
i = 0;  
error = 100;  
e = zeros(maxIter);  
while ((error > maxRelError) && (i < maxIter))  
funVal = func(start);  
i = i + 1;  
error = abs((funVal - start)/funVal) \* 100;  
e(1,i) = error;  
start = funVal;  
if(abs(f(funVal)) < convgCriteria)  
break;  
end  
end  
if i >= maxIter  
disp('Maximum Iteration number attained.');  
elseif error <= maxError  
disp('Convergence for maximum relative approximate error reached.');  
else  
disp('Convergence criteria for function value reached.');  
end  
figure  
subplot(2,1,2)  
plot (1:i, e(1:i))  
grid on;  
title('FixedPointError vs Iteration')  
print -djpg FixedPointError.jpg  
str = input('Input non-linear function: ','s');

func = str2func(['@(x)' str]);  
str1 = input('Enter f`(x): ','s');  
func1 = str2func(['@(x)' str1]);  
init = input('Enter initial guess: ');  
maxRelError = input('Enter the maximum allowed relative approximate error(in %): ');  
convgCriteria = input('Enter convergence criterion for function value: ');  
maxIter = input('Enter the maximum number of iterations: ');  
figure  
fplot(func,[(a-10) (a+10)]); % function plot  
grid on;  
title('func(x)vs x')  
print -djpg NewtonRaphsonFunc.jpg  
i = 0;  
error = 100;  
while ((error > maxRelError) && (i < maxIter))  
next = start - (func(start)/func1(start));  
i = i + 1;  
error = abs((next - start)/next) \* 100;  
e(1,i)= error;  
start = next;  
if(abs(func(next)) < convgCriteria)  
break;  
end  
end  
if i >= maxIter  
disp('Maximum Iteration number attained.');  
elseif error <= maxRelError  
disp('Convergence for maximum relative approximate error reached.');  
else  
disp('Convergence criteria for function value reached.');  
end  
figure  
subplot(2,1,2)  
plot (1:i,e(1:i)) % error plot  
grid on;  
title('NewtonRaphsonError vs Iteration')  
print -djpg NewtonRaphsonError.jpg  
str = input('Input non-linear function in x: ','s');  
func = str2func(['@(x)' str]);  
small = input('Enter smaller starting point: ');  
large = input('Enter larger starting point: ');  
maxRelError = input('Enter the maximum allowed relative approximate error(in %): ');  
convgCriteria = input('Enter convergence criterion for function value: ');  
maxIter = input('Enter the maximum number of iterations: ');

figure  
fplot(func,[(small - 10) (large + 10)]);  
grid on;  
title('func(x)vs x')  
print -djpg SecantFun.jpg  
i = 0;  
error = 100;  
while((error > maxRelError) && (i < maxIter))  
val = large - func(large) \* ((large - small)/(func(large) - func(small)));  
i = i + 1;  
error = abs((val - large)/val) \* 100;  
e(1,i)= error;  
small = large;  
large = val;  
if(abs(func(val)) < convgCriteria)  
break;  
end  
end  
if i >= maxIter  
disp('Maximum Iteration number attained.');  
elseif error <= maxRelError  
disp('Convergence for maximum relative approximate error reached.');  
else  
disp('Convergence criteria for function value reached.');  
end  
figure  
subplot(2,1,2)  
plot (1:i, e(1:i));  
grid on;  
title('SecantError vs Iteration')  
print -djpg SecantError.jpg