```
#include <iostream>
#include <stdlib.h>
#include <cmath>
using namespace std;

double fn(double x) {
	double result = pow(x,2) - (4 * x) + 4 - log(x);
	return result;

	//double result = exp(x) + pow(2, (-x)) + 2 * cos(x) - 6;
	//return result;

	//double result = -pow(x, 3) - cos(x);
	//return result;
}

double dfn(double x) {
	double result = exp(x) - pow(2, (-x)) * log(2) - 2 * sin(x);
	return result;
}



typedef double (* func)(double args);

int secantMethod(double p0, double p1, func fn, double TOL, int N0) {
	double p2 = 0;
	double q0 = fn(p0);
	double q1 = fn(p1);
	int i = 0;
	while (i < N0) {
		p2 = p1 - ((q1 * (p1 - p0))/(q1 - q0));
		cout << "Iteration " << i << ": " << p2 << endl;
		if (abs(p2 - p1) <= TOL) return p2;

		p0 = p1;
		q0 = q1;
		p1 = p2;
		q1 = fn(p1);

		i++;
	}
	return p2;
}
```

```
int newtonMethod(double p0, func fn, func dfn, double TOL, int N0) {
        int i = 0;
        double p1 = 0;

        while (i < N0) {
                //cout << p0 << endl;
                double f = fn(p0);
                double df = dfn(p0);

                p1 = p0 - (f / df);

                cout << "Iteration " << i << ": " << p1 << endl;
                if (abs(p1 - p0) < TOL) return p1;

                p0 = p1;

                i++;
        }
}

int main () {
        secantMethod(2, 4, (func)fn, .0000001, 100);
        newtonMethod(1, (func) fn, (func) dfn, .000001, 100);
        return 0;
}
```

Newton's Method:

```
Iteration 0: 3.4698
Iteration 1: 2.72613
Iteration 2: 2.19729
Iteration 3: 1.91427
Iteration 4: 1.835
Iteration 5: 1.82941
Iteration 6: 1.82938
Iteration 7: 1.82938
```

Secant method:

```
Iteration 0: 2.41922
Iteration 1: 2.75604
Iteration 2: 3.31702
Iteration 3: 3.00977
Iteration 4: 3.05067
Iteration 5: 3.05729
Iteration 6: 3.0571
Iteration 7: 3.0571
Iteration 8: 3.0571
```