```go
package main

import (
    "fmt"  // Println関数
    "math" // SmallestNonzeroFloat64定数,
)

func main() {
    // math.SmallestNonzeroFloat64は、
    // 4.940656458412465441765687928682213723651e-324
    // を表す定数であり、Goのfloat64型の値の中で絶対値が最小のものである
    fmt.Println("f1")
    newton(f1, f1_prime, math.SmallestNonzeroFloat64, 0.1)
    fmt.Println("f2")
    newton(f2, f2_prime, math.SmallestNonzeroFloat64, 0.1)
}

// newton(f, f_prime, e, x0)の引数
//   f       : 対象関数
//   f_prim  : fの一次導関数
//   e       : "|x_(k+1) - x_k| < e && f(x_(k+1)) < e" が満たされたときに終了する
//   x0      : x0
// 表示される値
//   k    : 終了条件が満たされた、最初のkの値
//   x    : x_kの値
//   f(x) : f(x_k)の値
// 挙動の解説
//   1. "|x_(k+1) - x_k| < e && f(x_(k+1)) < e" が初めて満たされたときに、その時点でのk, x, f(x)を終了する。
//   2. "|x_(k+1) - x_k| < e"が満たされたとき、その時点でのk, x, f(x)を表示する。また、表示される行の末尾に":: delta x < e satisfied"が追加される。
//   3. "|f(x_(k+1))| < e"が満たされたとき、その時点でのk, x, f(x)を表示する。また、表示される行の末尾に":: f(x) < e satisfied"が追加される。
// なお、ステップの実行回数は10回までとした。
func newton(f func(float64) float64, f_prime func(float64) float64, e float64, x0 float64) {
    /*
        各ステップでのx_kを表しているのがx_prev変数
        各ステップでのx_k+1を表しているのがx_next変数
    */
    var x_prev float64
    var x_next float64

    k := 0
    x_prev = x0
    for k < 10 {
        // 修正量を計算し、それから反復列の次の近似解を求める
        x_next = x_prev - f(x_prev)/f_prime(x_prev)

        // 途中経過を表示する
        if math.Abs(f(x_next)) < e {
            // k, x, f(x)を表示する
            fmt.Printf("k=%v, x=%v, f(x)=%v :: f(x) < e satisfied\n", k, x_next, f(x_next))
        }
        if math.Abs(x_next-x_prev) < e {
            // k, x, f(x)を表示する
            fmt.Printf("k=%v, x=%v, f(x)=%v :: delta x < e satisfied\n", k, x_next, f(x_next))
        }

        // 終了条件を両方共満たしたとき終了する
        if math.Abs(x_next-x_prev) < e && math.Abs(f(x_next)) < e {
            break
        }
        x_prev = x_next
        k++
    }
    // 終了時のk, x, f(x)を表示する
    fmt.Printf("k=%v, x=%v, f(x)=%v :: finish\n", k, x_next, f(x_next))
}

// 問に挙げられた方程式のうち、4次方程式の方を表す関数
// xを受け取り、関数を評価した値を返す
func f1(x float64) float64 {
    return -2.2*math.Pow(x, 4) + 3.5*math.Pow(x, 3) + 4.1*math.Pow(x, 2) + 3.3*x - 2.7
```

```
71 }
72
73 // f1の一次導関数を表す
74 func f1_prime(x float64) float64 {
75     return -8.8*math.Pow(x, 3) + 10.5*math.Pow(x, 2) + 8.2*x + 3.3
76 }
77
78 // 問に挙げられた方程式のうち、余弦関数を含む方を表す関数
79 // xを受け取り、関数を評価した値を返す
80 func f2(x float64) float64 {
81     return -3*math.Cos(2*x+2) + math.Exp(x+1) - 2*x - 30
82 }
83
84 // f2の一次導関数を表す
85 func f2_prime(x float64) float64 {
86     return 6*math.Sin(2*x+2) + math.Exp(x+1) - 2
87 }
```