```go
/*
 * Carl Kingsford --- Oct 6, 2014
 *
 * This code was written on the fly during class. As such it isn't as polished
 * as one would hope; I've left it this way so that you can see how a program
 * looks after it's been written quickly and editted a bit.
 */
package main

import (
    "fmt"
    "strconv"
    "os"
    "math"
)

// Return the Rhs for a given Lhs
func getRhsFor(char string, lhs, rhs []string) (string, bool) {
    for i, l := range lhs {
        if l == char {
            return rhs[i], true
        }
    }
    return "", false
}

// applyRules will take string "start" and apply each of the given rules in
// parallel, returning the new string.
func applyRules(lhs, rhs []string, start string) string {
    var out string
    for _, char := range start {
        charRhs, existed := getRhsFor(string(char), lhs, rhs)
        if existed {
            out = out + charRhs
        } else {
            out = out + string(char)
        }
    }
    return out
}


/*
    A and B: draw line forward in the direction you¿re facing
-: turn right by 60°
+: turn left by 60°
*/
func drawLindenmayer(s string) {
    const w = 10000
    const h = 10000
    pic := CreateNewCanvas(w, h)
    pic.SetStrokeColor(MakeColor(255, 0, 255))
    pic.SetLineWidth(1)
    x,y := 0.0, 0.9*h
    dir := 0.0
    step := 2.0

    pic.MoveTo(x,y)

    for _, c := range s {
        if c == 'A' || c == 'B' {
            x = x + step * math.Cos(dir)
            y = y - step * math.Sin(dir)
            pic.LineTo(x, y)

        } else if c == '+' {
            // turn left
            dir = dir + math.Pi / 3.0
        } else if c == '-' {
            // trun right
            dir = dir - math.Pi / 3.0
        } else {
            panic("Wow, somethings really wrong.")
        }
    }
    pic.Stroke()

    pic.SaveToPNG("Lind.png")
}

//===== A FLOAT STACK ====

type Pen struct {
    x, y float64
    dir float64
}

func createPenStack() []Pen {
   return make([]Pen, 0)
}

func pushPen(S []Pen, item Pen) []Pen {
   return append(S, item)
}

func popPen(S []Pen) ([]Pen, Pen) {
   if len(S) == 0 {
       panic("Can¿t pop empty stack!")
   }
   item := S[len(S)-1]
   S = S[0:len(S)-1]
   return S, item
}


// Draw the Plant Lindenmayer System
func drawPlant(s string) {
    const w, h = 10000, 10000
    pic := CreateNewCanvas(w, h)
    pic.SetStrokeColor(MakeColor(255, 0, 255))
    pic.SetLineWidth(1)

    var myPen = Pen{x:0.5*w, y: 0.5*h, dir:math.Pi/2}
    step := 10.0


    penStack := createPenStack()

    pic.MoveTo(myPen.x,myPen.y)
    for _, c := range s {
        switch c {
        case 'F':
            myPen.x = myPen.x + step * math.Cos(myPen.dir)
            myPen.y = myPen.y - step * math.Sin(myPen.dir)
            pic.LineTo(myPen.x, myPen.y)

        case '+':
            // turn left
            myPen.dir = myPen.dir + math.Pi * (25.0 / 180.0)

        case '-':
            // trun right
            myPen.dir = myPen.dir - math.Pi * (25.0 / 180.0)

        case '[':
            // save
            penStack = pushPen(penStack, myPen)

        case ']':
            // restore
            penStack, myPen = popPen(penStack)
            pic.MoveTo(myPen.x,myPen.y)

        case 'X':

        default:
            panic("Wow, somethings really wrong.")
        }
    }
    pic.Stroke()
    pic.SaveToPNG("Plant.png")
}

func main() {
    lhs := []string{"X", "F"}
    rhs := []string{"F-[[X]+X]+F[+FX]-X", "FF"}

    fmt.Println(lhs, rhs)

    if len(os.Args) < 2 {
        fmt.Println("Wrong.")
        return
    }

    steps, err := strconv.Atoi(os.Args[1])
    if err != nil {
        fmt.Println("Wrong.")
        return
    }

    var current string = "X"
    for i := 0; i < steps; i++ {
        current = applyRules(lhs, rhs, current)
    }
    fmt.Println(current)

    drawPlant(current)
}
```