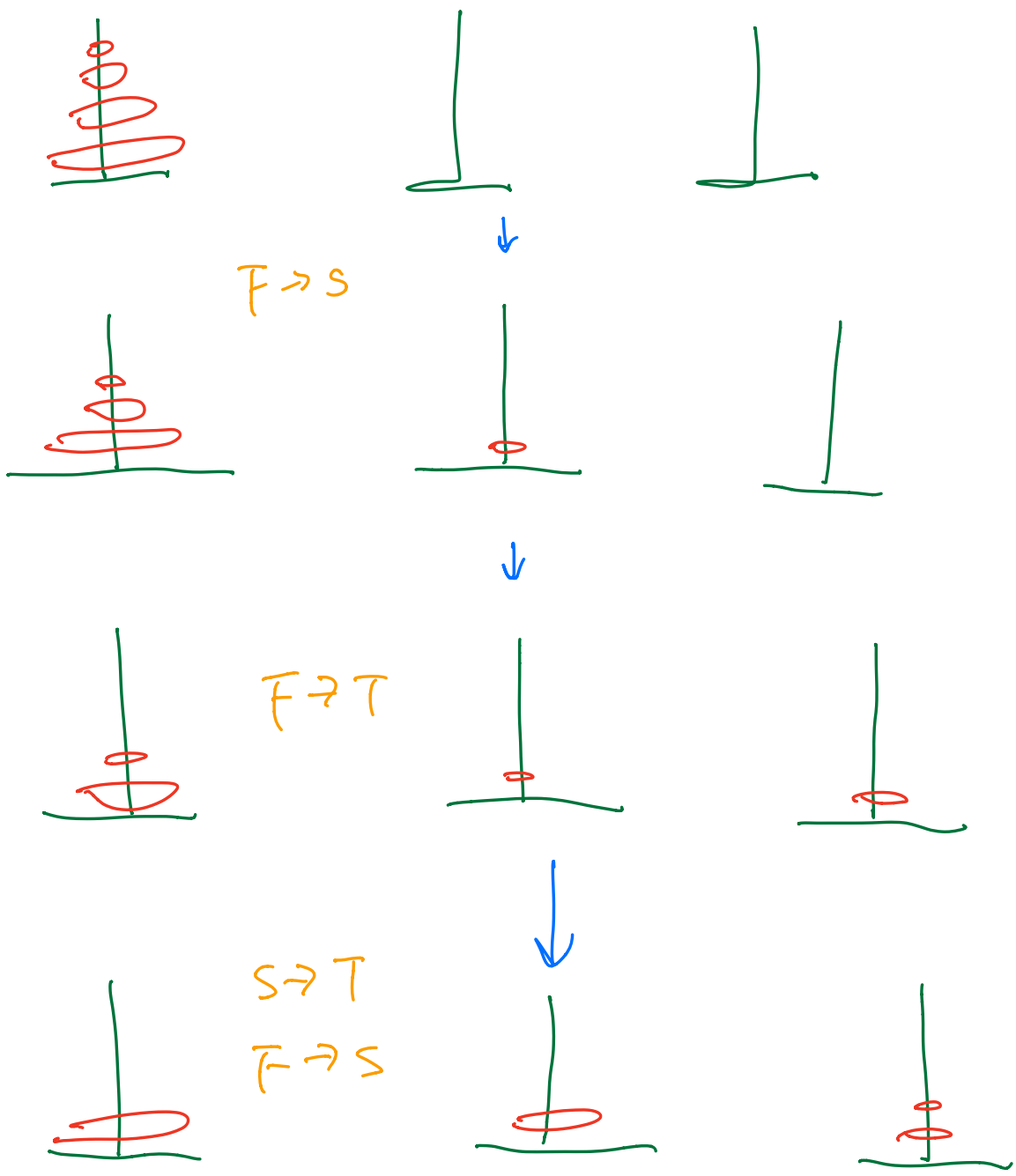
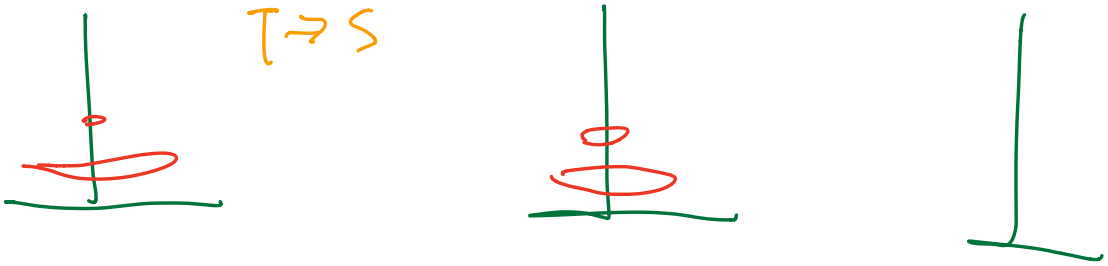
Assignment 3 Design.pdf.



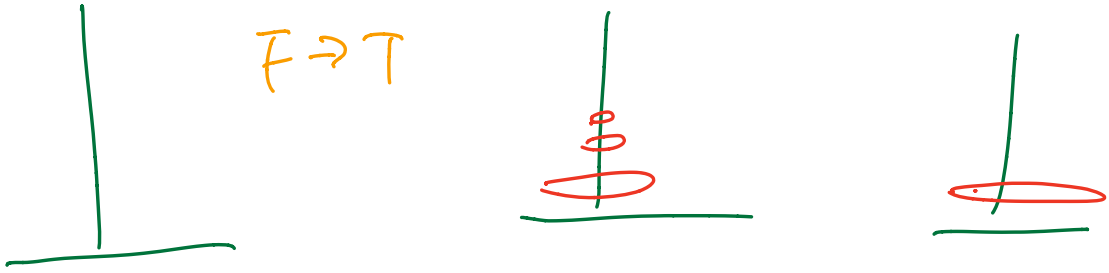
T → F

T → S



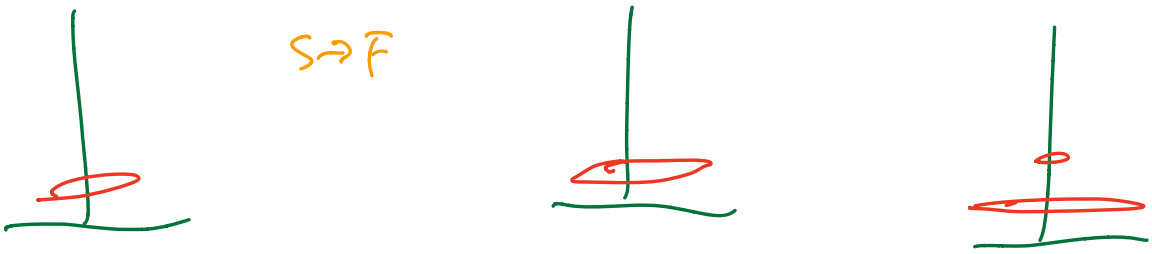
F → S

F → T



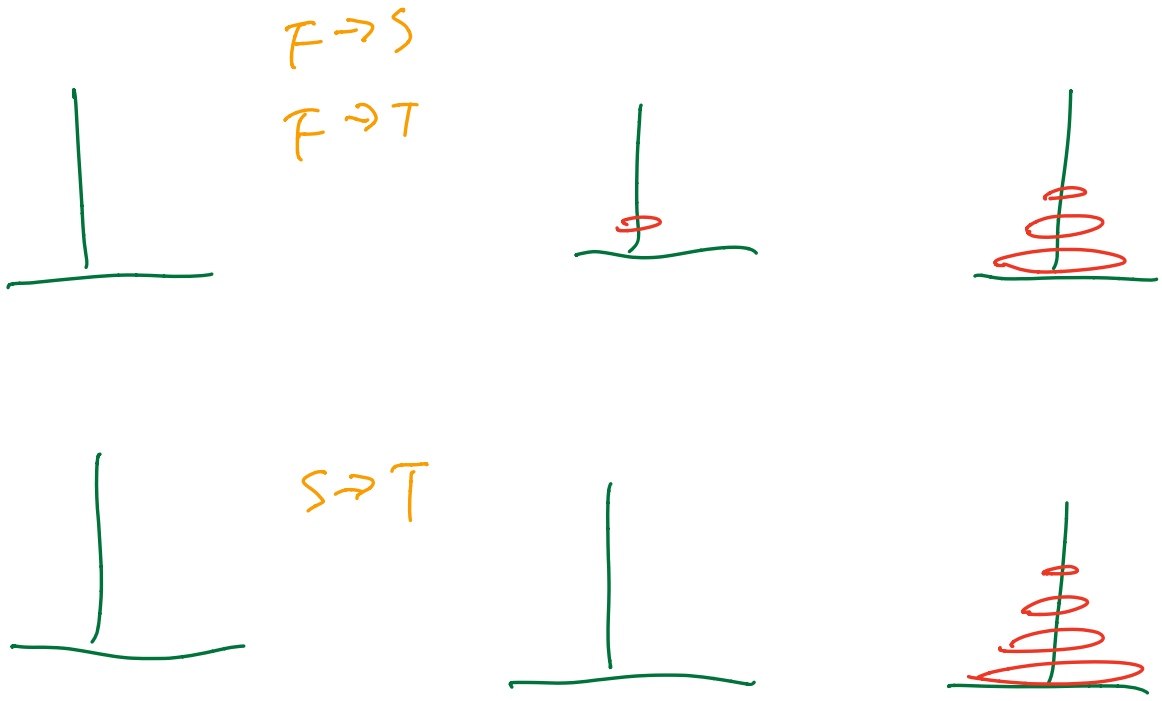
S → T

S → F



T → F

S → T



Tower.c

take user input -s
-t
-n

for -n !, set stack number,
if nothing input, set to 5

will use option ":stn:"

case ':':

set to 5.

case 'n'

set to atoi "optarg"

Recursion:

take parameter count

to keep track on how

many moves.

int recursion (int count, int size)

↑

return the count.

int Recursion (size, from, to, stop)

if (size == 1)

move from, to.

return count ++

Recursion (size-1, from, stop, to)

move n to to.

Recursion (size-1, stop, to, from)

```
Move disk 1 from peg B to peg C
Move disk 2 from peg B to peg A
Move disk 1 from peg C to peg A
Move disk 4 from peg C to peg B
Move disk 1 from peg A to peg B
Move disk 2 from peg A to peg C
Move disk 1 from peg B to peg C
Move disk 3 from peg A to peg B
Move disk 1 from peg C to peg A
Move disk 2 from peg C to peg B
Move disk 1 from peg A to peg B

Number of moves: 31
```

```
TheMachine:tower darrell$ ./tower -s -n 3
================================
----------    STACKS    ----------
================================
Move disk 1 from peg A to peg B
Move disk 2 from peg A to peg C
Move disk 1 from peg B to peg C
Move disk 3 from peg A to peg B
Move disk 1 from peg C to peg A
Move disk 2 from peg C to peg B
Move disk 1 from peg A to peg B
```

```
TheMachine:tower darrell$ ./tower -s -r
================================
----------    STACKS    ----------
================================
Move disk 1 from peg A to peg B
Move disk 2 from peg A to peg C
Move disk 1 from peg B to peg C
Move disk 3 from peg A to peg B
Move disk 1 from peg C to peg A
Move disk 2 from peg C to peg B
Move disk 1 from peg A to peg B
Move disk 4 from peg A to peg C
Move disk 1 from peg B to peg C
Move disk 2 from peg B to peg A
Move disk 1 from peg C to peg A
Move disk 3 from peg B to peg C
```

Stack implementation.

F = from T = To S = stop

1 Disk.

F → T

2 Disks

F → S

F → T

S → T

3 Disks

F → T 1

F → S 2

T → S 3

F → T 1

S → F 2

S → T 3

F → T 1

4 Disks

F → S 1

F → T 2

S → T 3

F → S 1

T → F 2

T → S 3

F → S 1

F → T 2

S → T 3

S → T

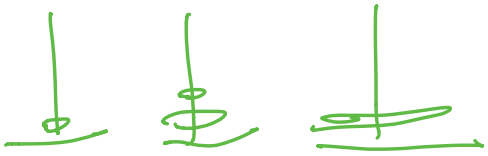
T → F

S → T

F → S

F → T

S → T



Move = $n^2 - 1$

If even

F ⇔ S

F ⇔ T

S ⇔ T

If odd

F ⇔ T

F ⇔ S

T ⇔ S

Pseudocode

```
while ( destination not full ) {
    if even
        F ⇔ S      // check which
        F ⇔ T         one is legal
        S ⇔ T
    if odd
        F ⇔ T      // check which
        F ⇔ S         one is legal
        T ⇔ S
```

```
move ++  // To keep check
            the move
```