Lecture 9 Recursions

Prefix expression evaluation

Hanoi Tower Solving

Waiting time of Run pattern

Outlines

- Stack
  - Simulation
  - Evaluation of Prefix Expressions
- Multiple stacks
  - Simulation
  - Hanoi Tower Panel
  - Auto-play

STACK

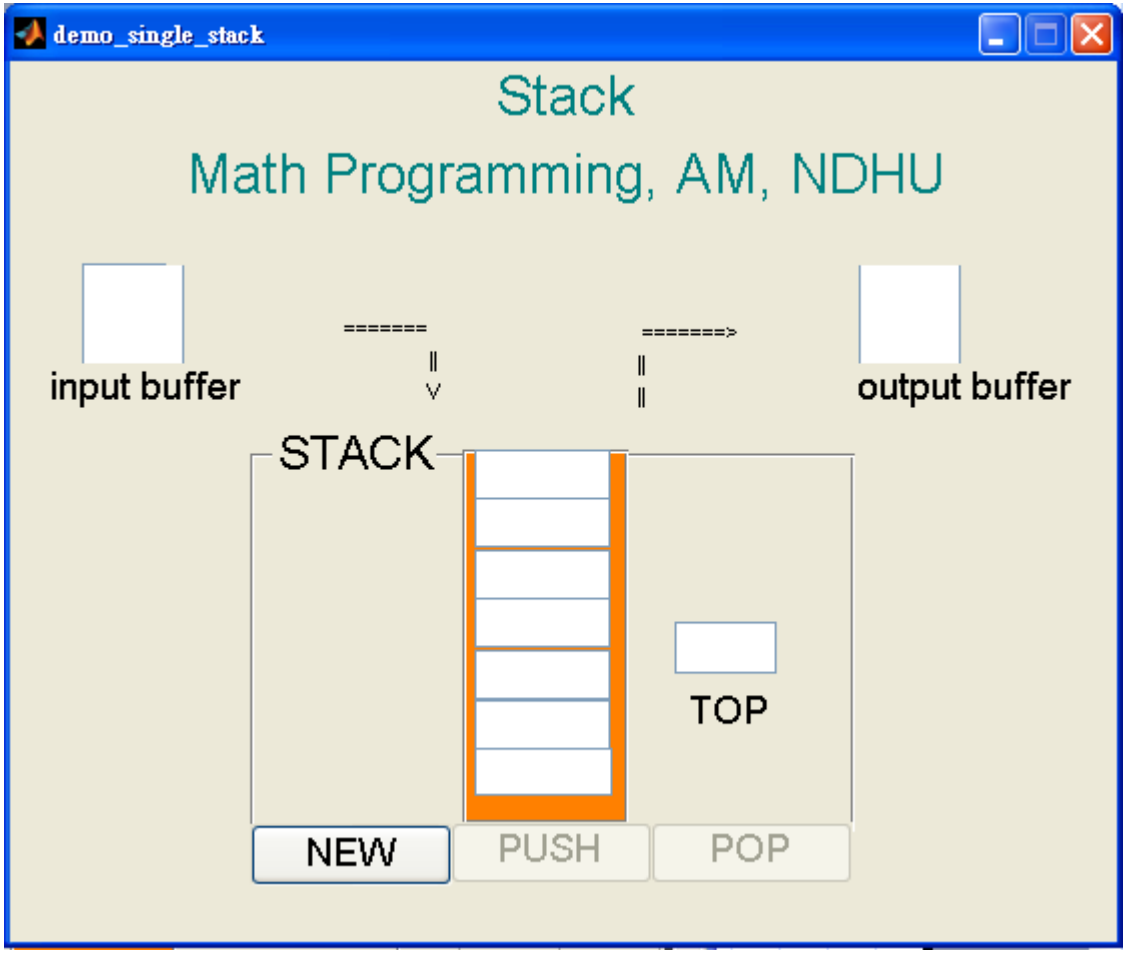
- Data structure
- Operations
  - Allocate an empty stack
  - Check empty
  - Push
  - Pop

Data Structure

- Two fields
  - An array
  - Top

Simulation of a stack

demo_single_stack.m
demo_single_stack.fig


demo_single_stack
Stack
Math Programming, AM, NDHU
input buffer
output buffer
STACK
TOP
NEW
PUSH
POP

New

- Function head: stk=new_stack()
- Function body:

```
stk.s=[];
stk.top=0;
```

Pop

- Function head: c=pop_stack(stk)
- Function body:

```
If stk.top > zero
        j = stack.top
        c =stk.s(j)
        stack.top = stack.top -1
else
    c = -1
end
```

Push

- Function head: stk=push(c,stk)
- Function body:

  stk.s = [stk.s c];

  stk.top = stk.top + 1

Is_empty

- Function head: flag=is_empty (stk)
- Function body:

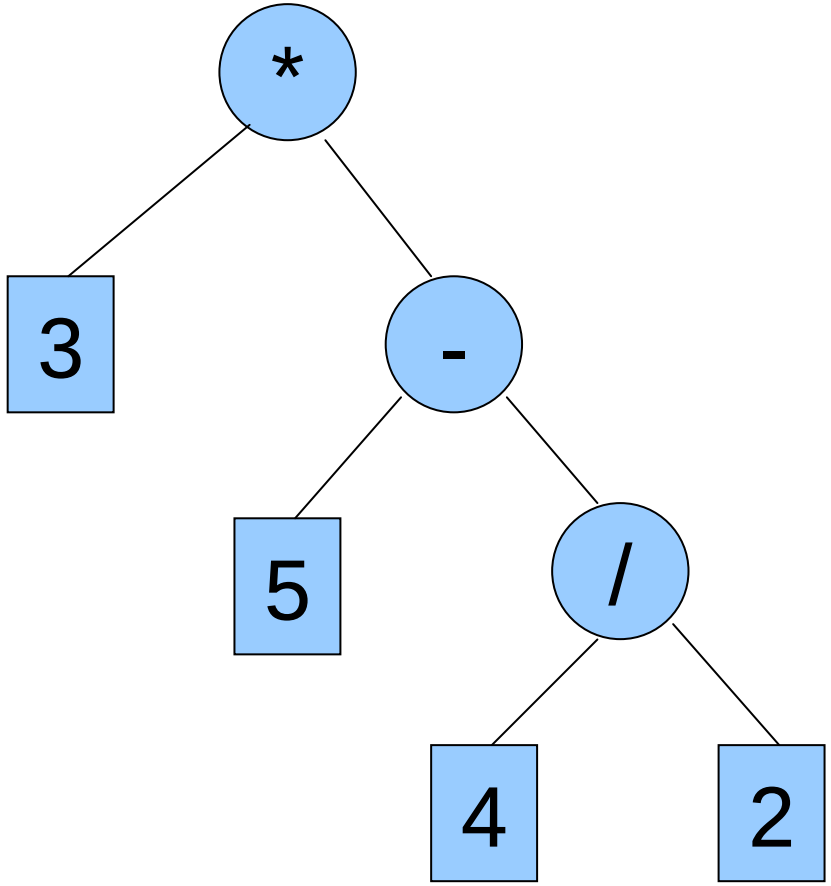
```
If stk.top==0
    flag = 1
else
    flag = 0;
end
```

Application

Evaluation of prefix notations

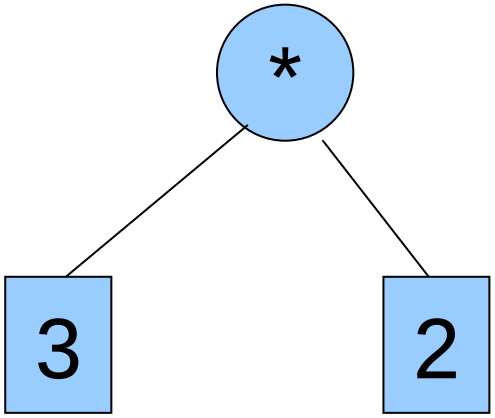
Binary tree

- V=3*(5-4/2)

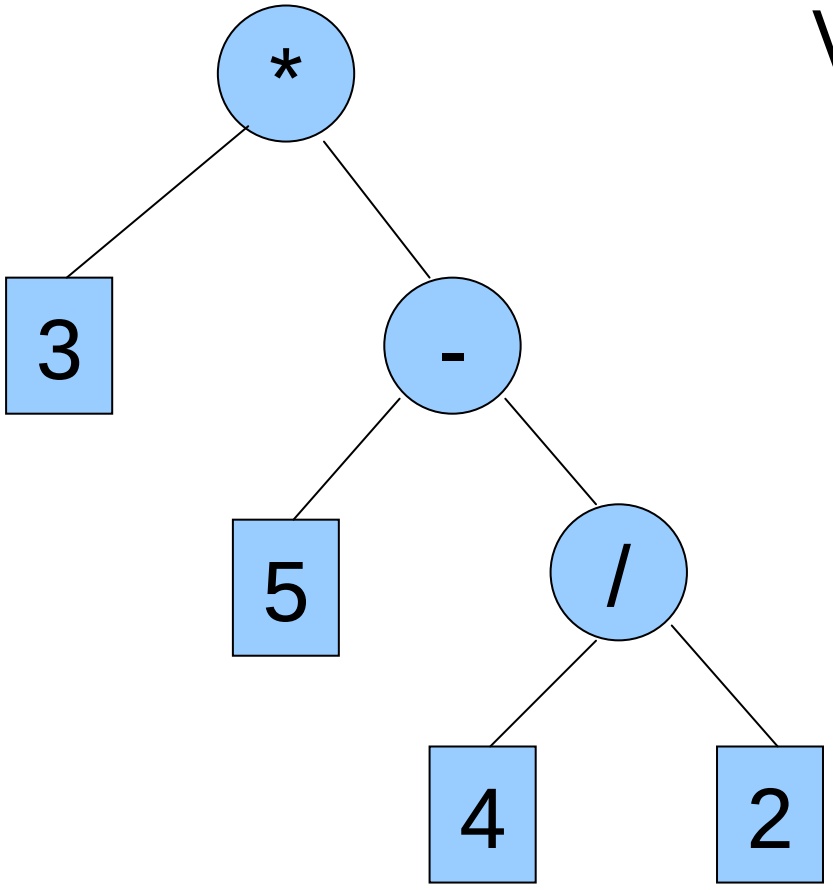


Prefix notations

- The operator is prefixed to two operands in an expression
- *32

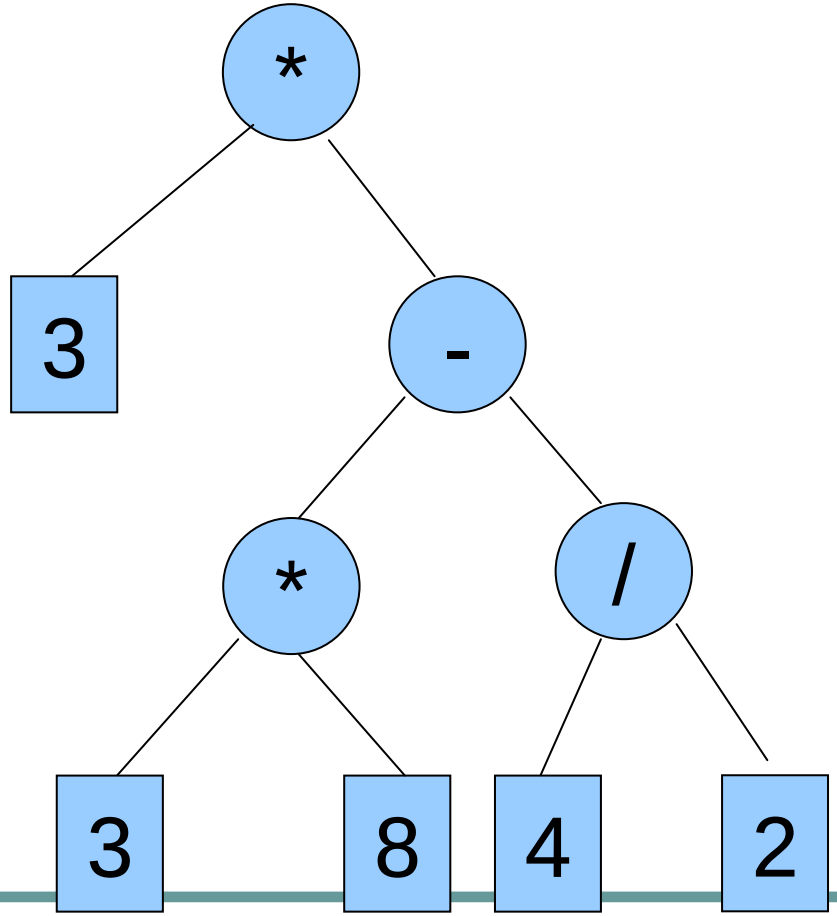


Example



V=*3-5/42

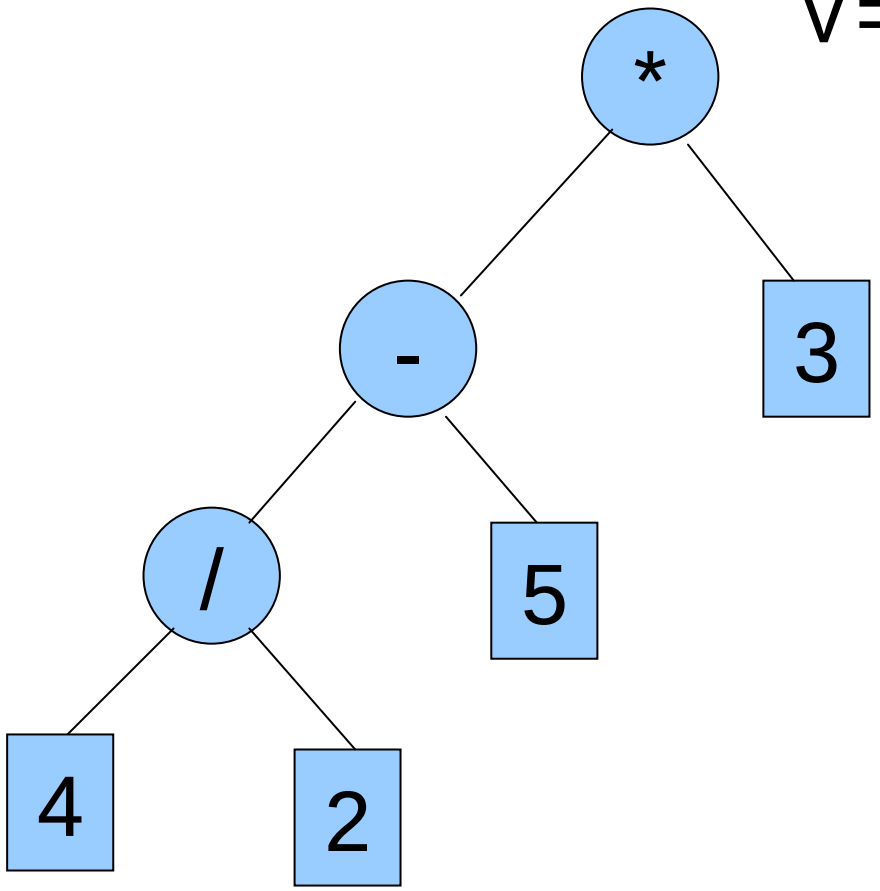
Prefix notations



V=*3-*38/42

Example

V=*-/4253

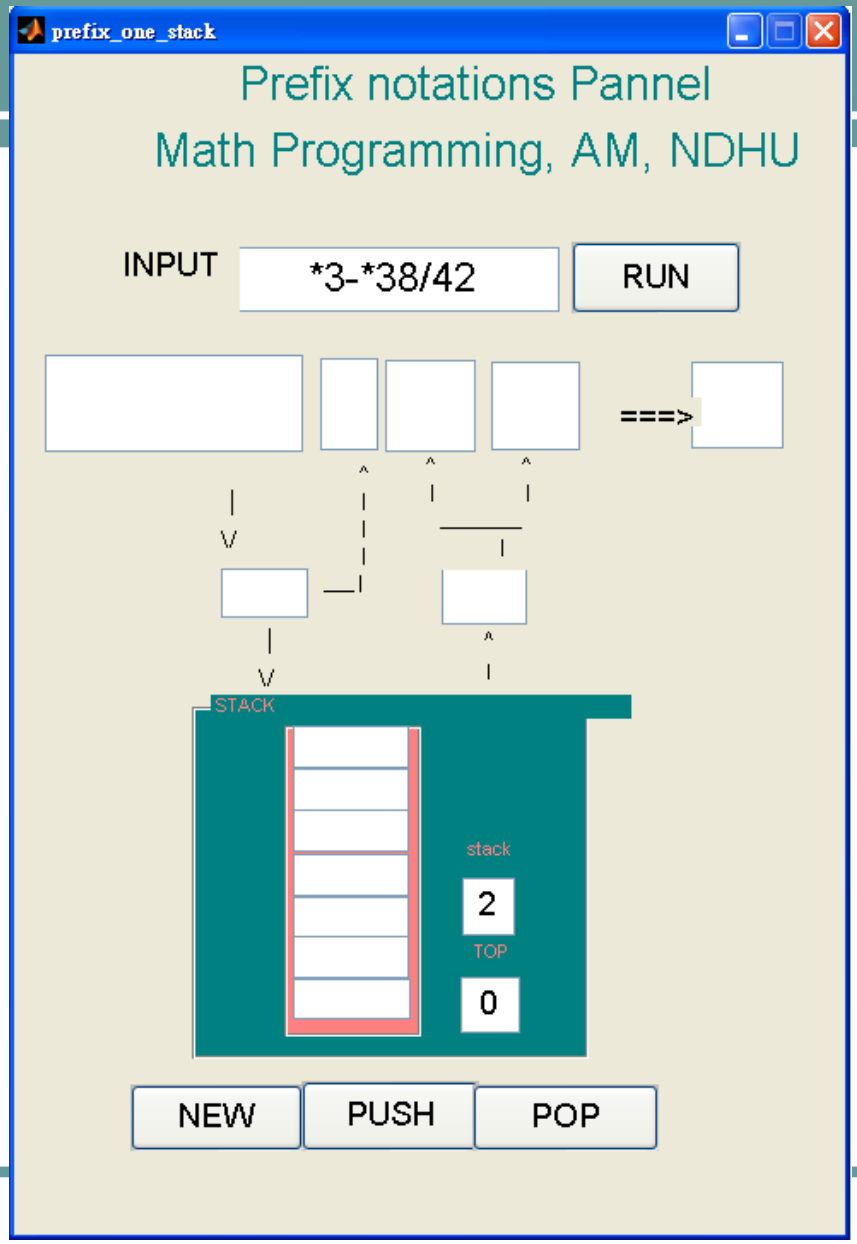


A simulator

- Prefix notations simulated here are constrained by
  - Operands in an expression are positive integers within {1,2,…,9}
  - Operators are within {+,-,*,/}
  - /ab is realized by floor(a/b)

Evaluation of Prefix notations

prefix_one_stack.m
prefix_one_stack.fig


prefix_one_stack
Prefix notations Pannel
Math Programming, AM, NDHU
INPUT
*3-*38/42
RUN
===>
STACK
stack
2
TOP
0
NEW
PUSH
POP

Exercise

Exercise2.pdf

Problem 7

- Apply an operator to two operands.

```
function v=evaluate(c,s1,s2)
        if c=='/'
            str=['v=floor(' s1 c s2 ')'];
        else
            str=['v=' s1 c s2];
        end
        eval(str);
```

Evaluation of a prefix expression

- Function head: v=prefix(ss)
- Function body
  - Allocate a stack, stk
  - n = length(ss)
  - For j=n:-1:1
    - c=ss(j)
    - If c is a digit, push c to stack stk.
    - If c is an operator,
      - If stk.top is greater than two
        - pop an element from stk and set it to d1
        - pop an element from stk and set it to d2
        - apply operand c to d1 and d2 and set the result to v
        - push v to stack stk
      - otherwise display(‘an invalid prefix expression’)

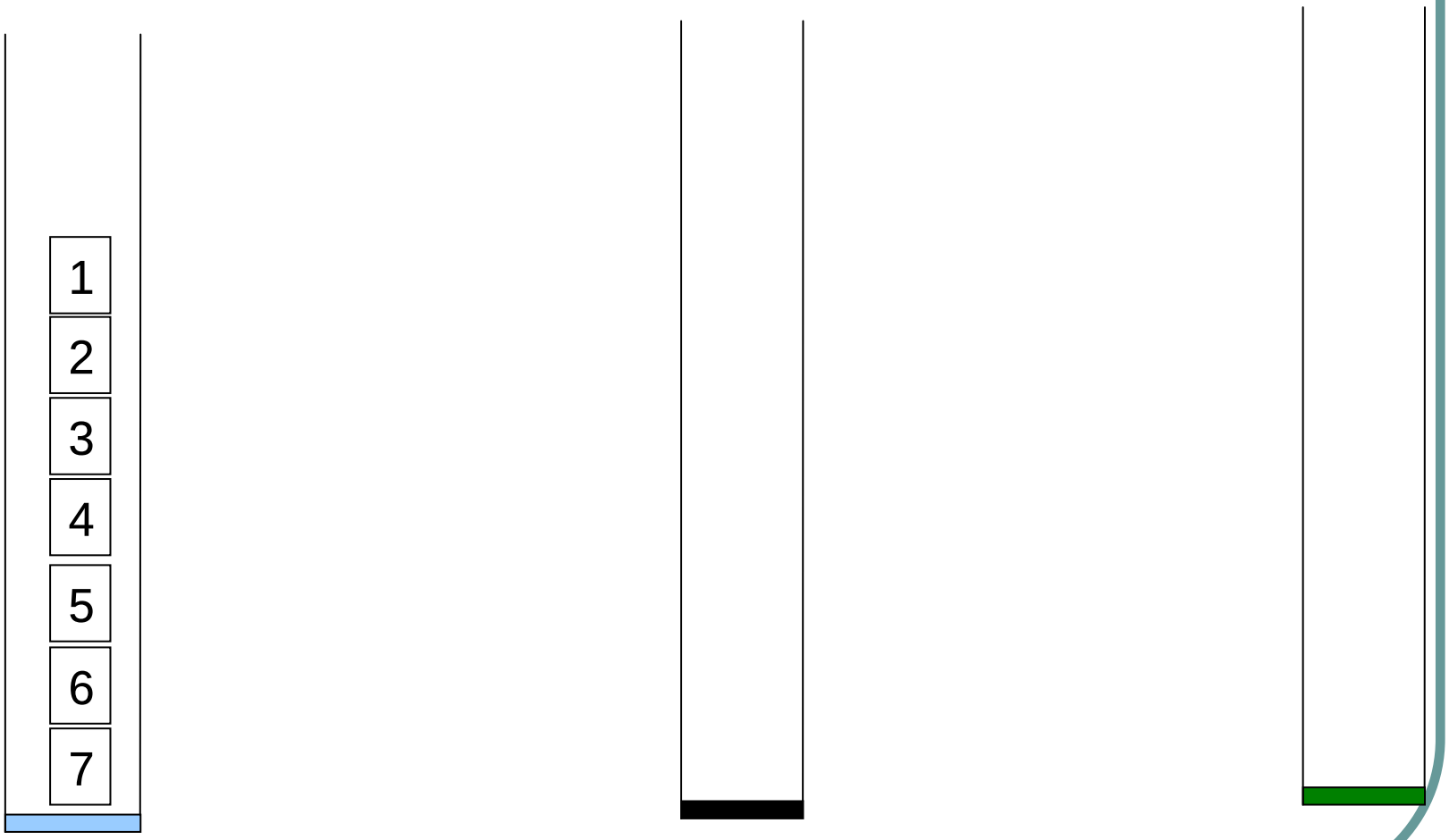
Multiple Stacks

Hanoi Tower Play Panel

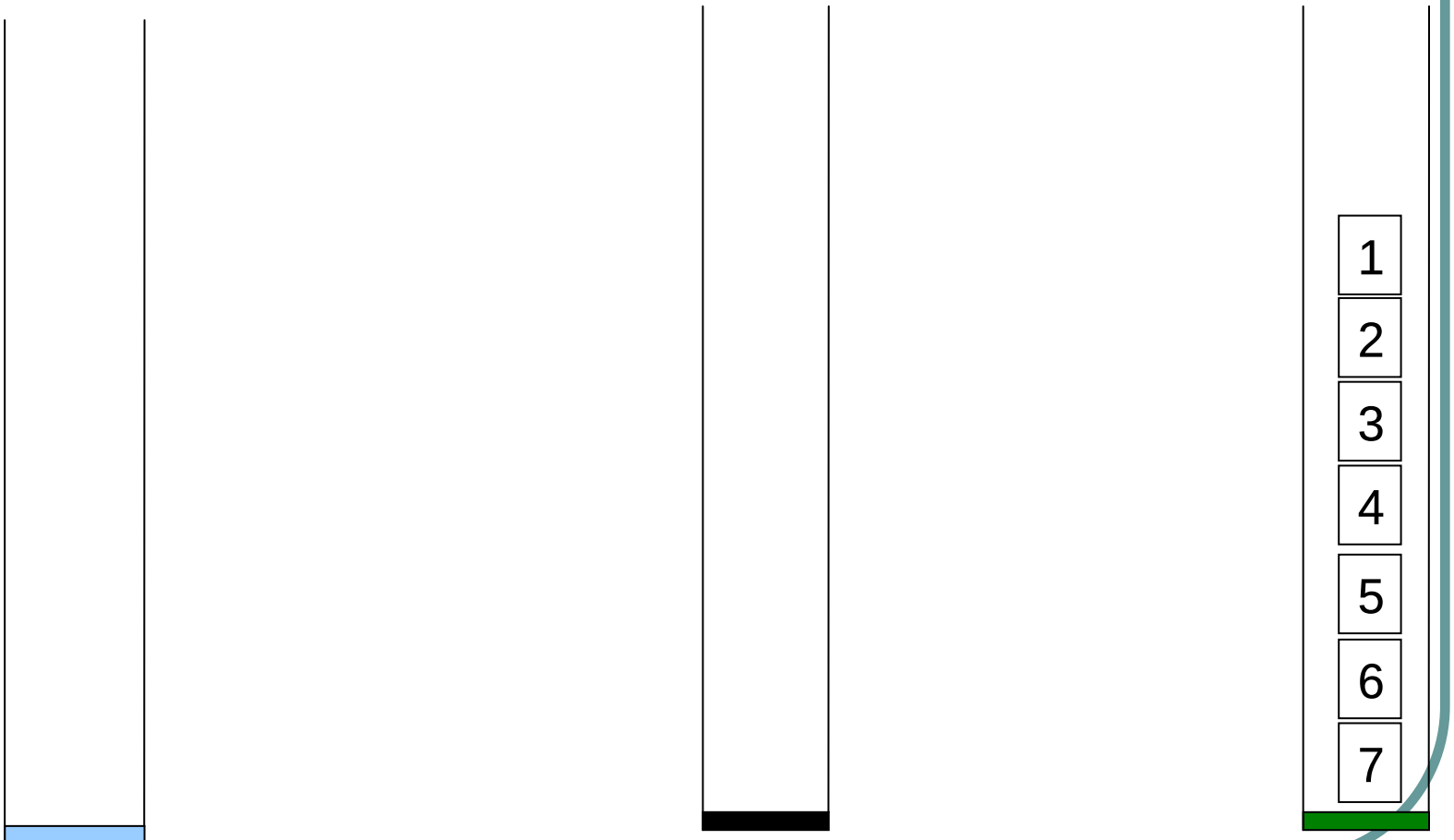
Problem statement

- Three stacks
- The first stack contains n disks, numbered from 1,…,n
- Move all disks from the first stack to the third stack.
- A disk is never on the top of a smaller disk

Initial state of the three stacks

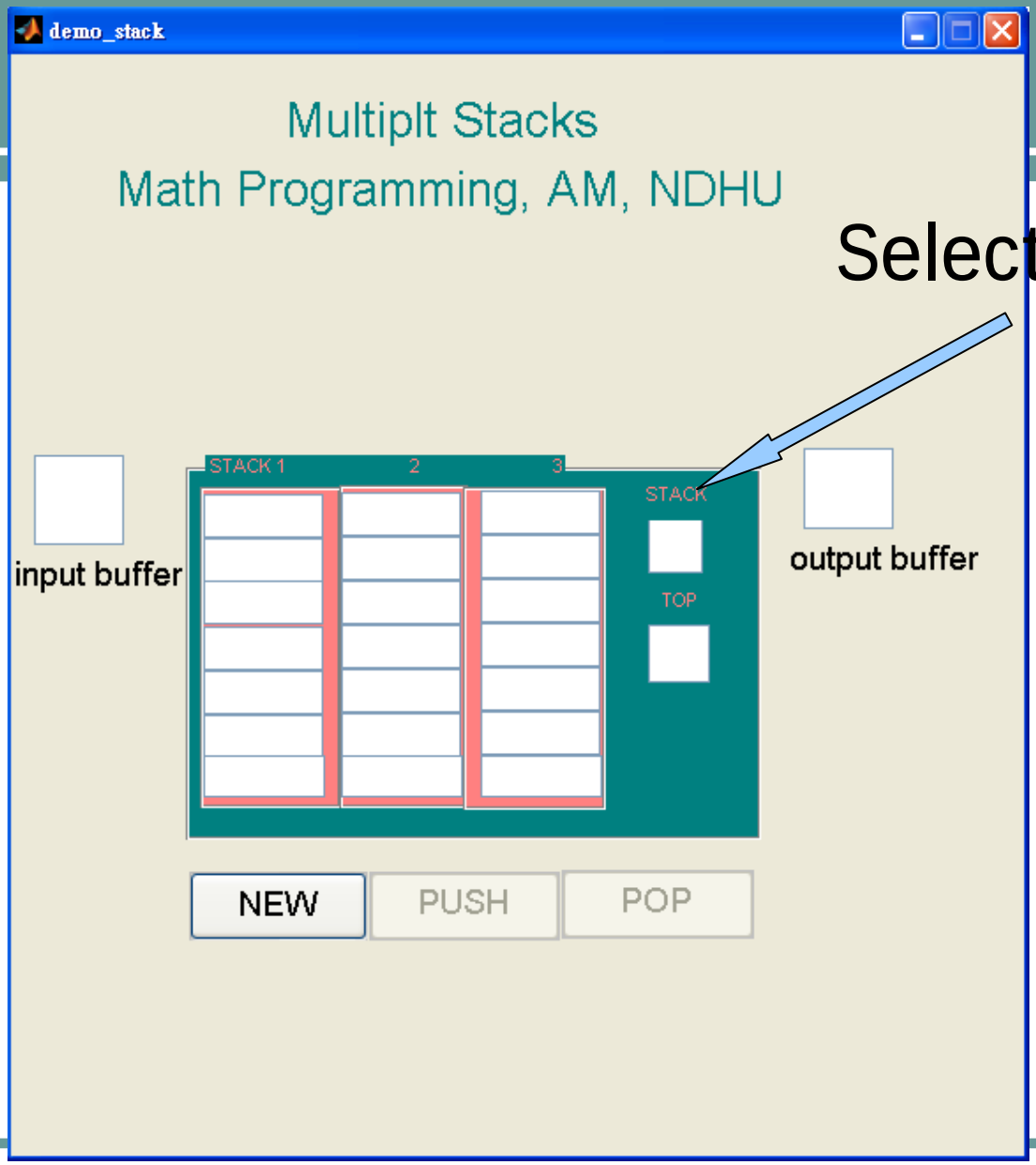


Final state



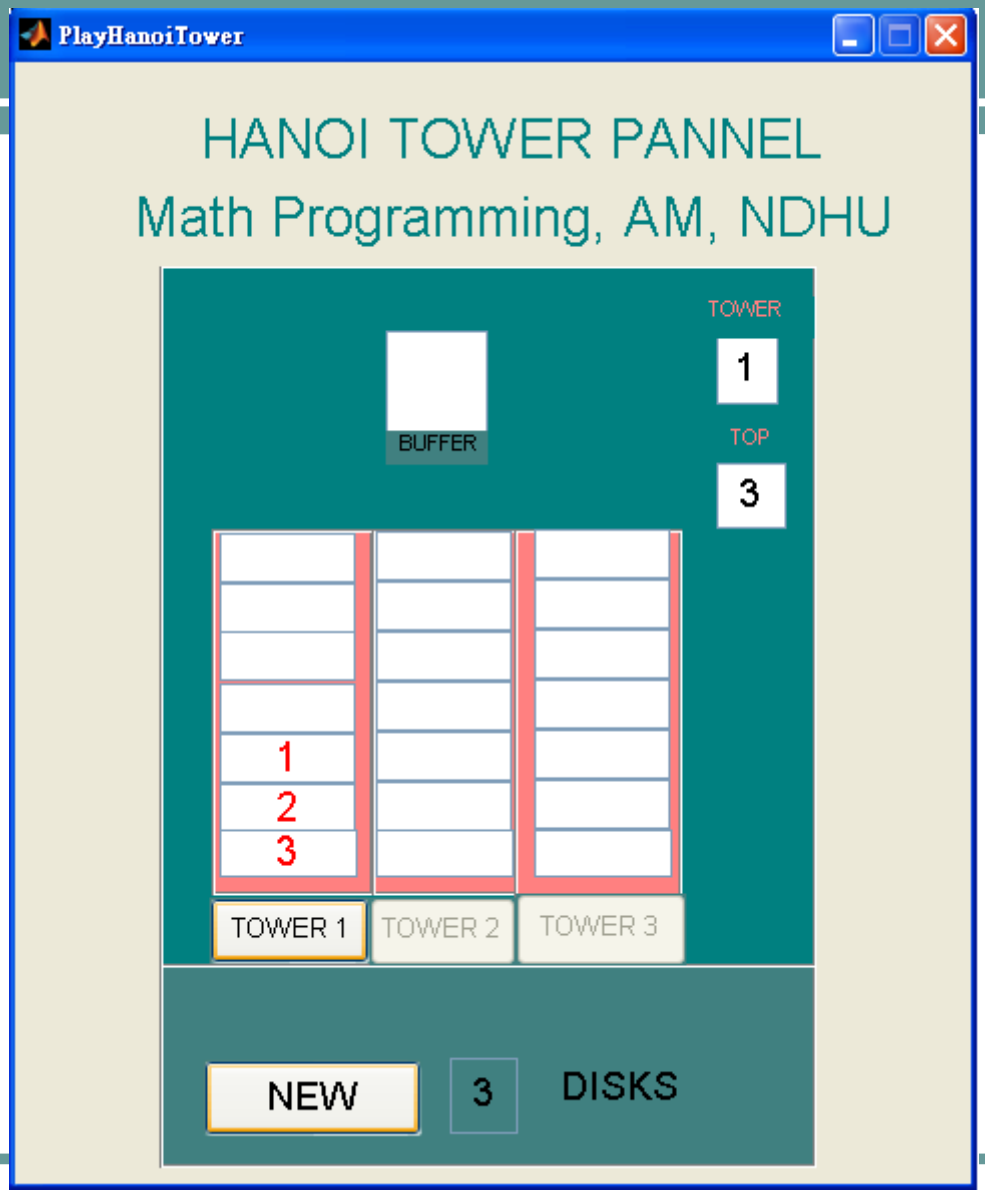
A GUI simulator of multiple stacks

demo_stack.m
demo_stack.fig


demo_stack
Multiplt Stacks
Math Programming, AM, NDHU
Select a stack
input buffer
STACK 1
2
3
STACK
TOP
output buffer
NEW
PUSH
POP

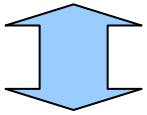
A Hanoi Tower Panel

PlayHanoiTower.m
PlayHanoiTower.fig

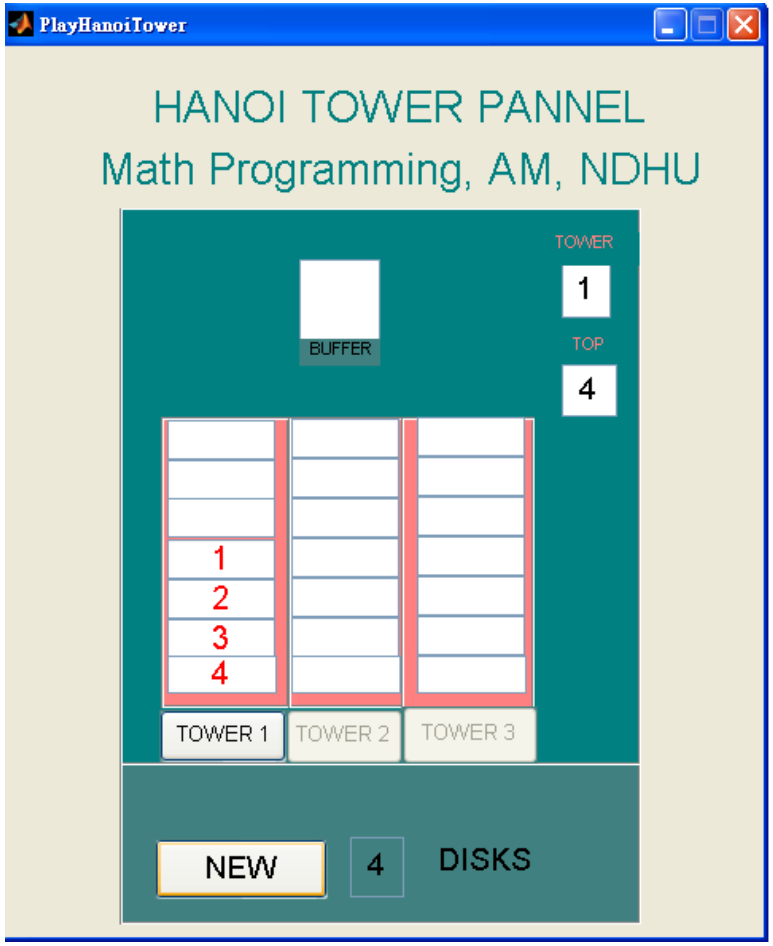

PlayHanoiTower
HANOI TOWER PANNEL
Math Programming, AM, NDHU
TOWER
1
BUFFER
TOP
3
1
2
3
TOWER 1
TOWER 2
TOWER 3
NEW
3
DISKS

Three steps for auto-play

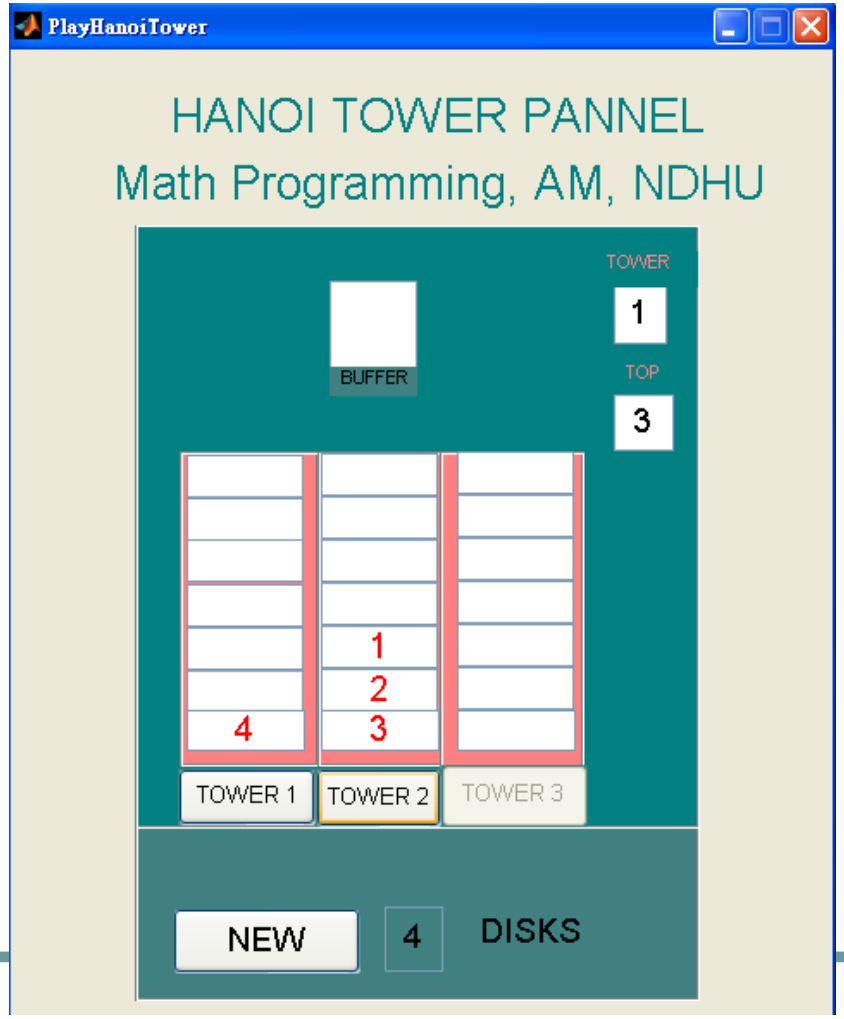
- Move n-1 objects from stack 1 to stack 2
- Move 1 object from stack 1 to stack 3
- Move n-1 objects from stack 2 to stack 3



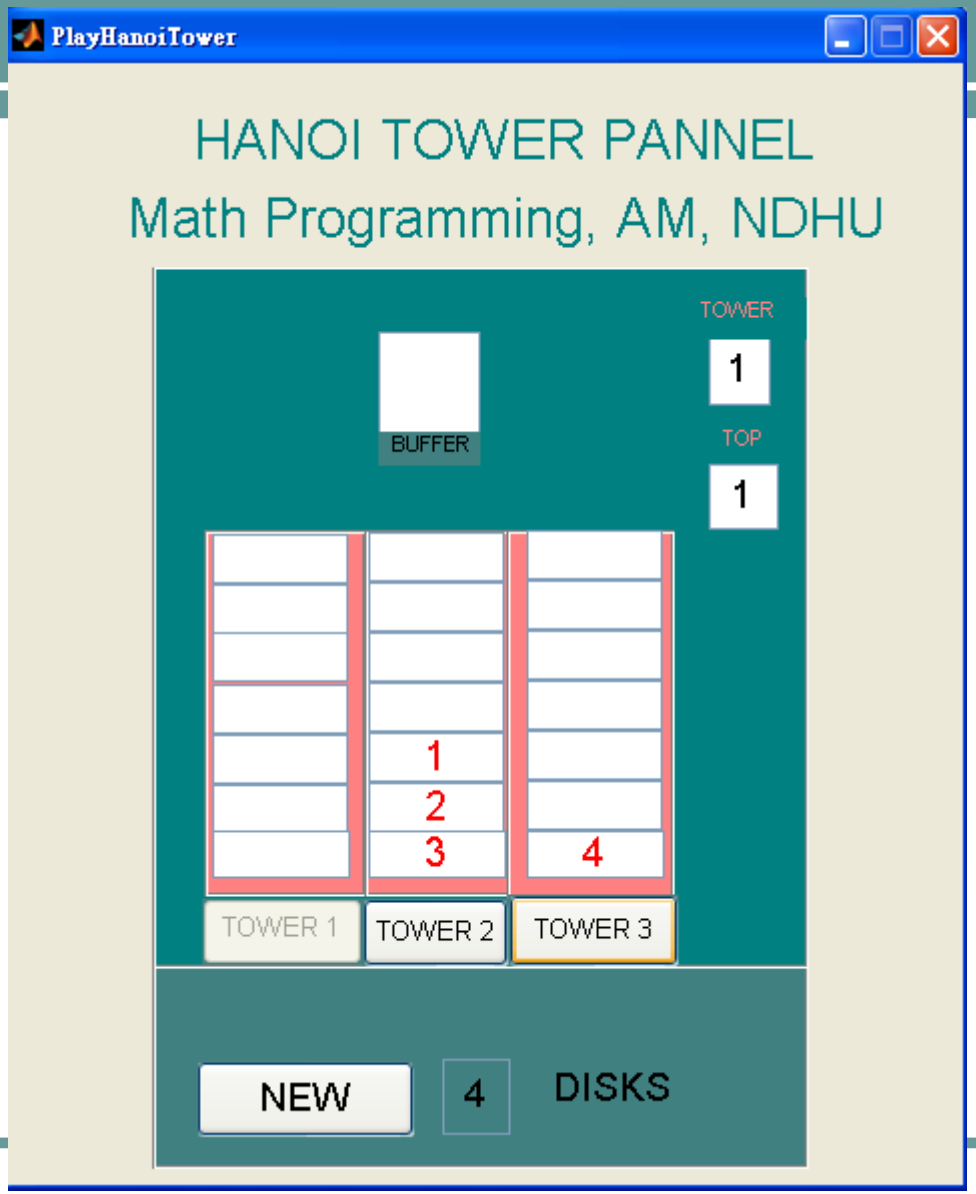
Initial state of 4-disk problem



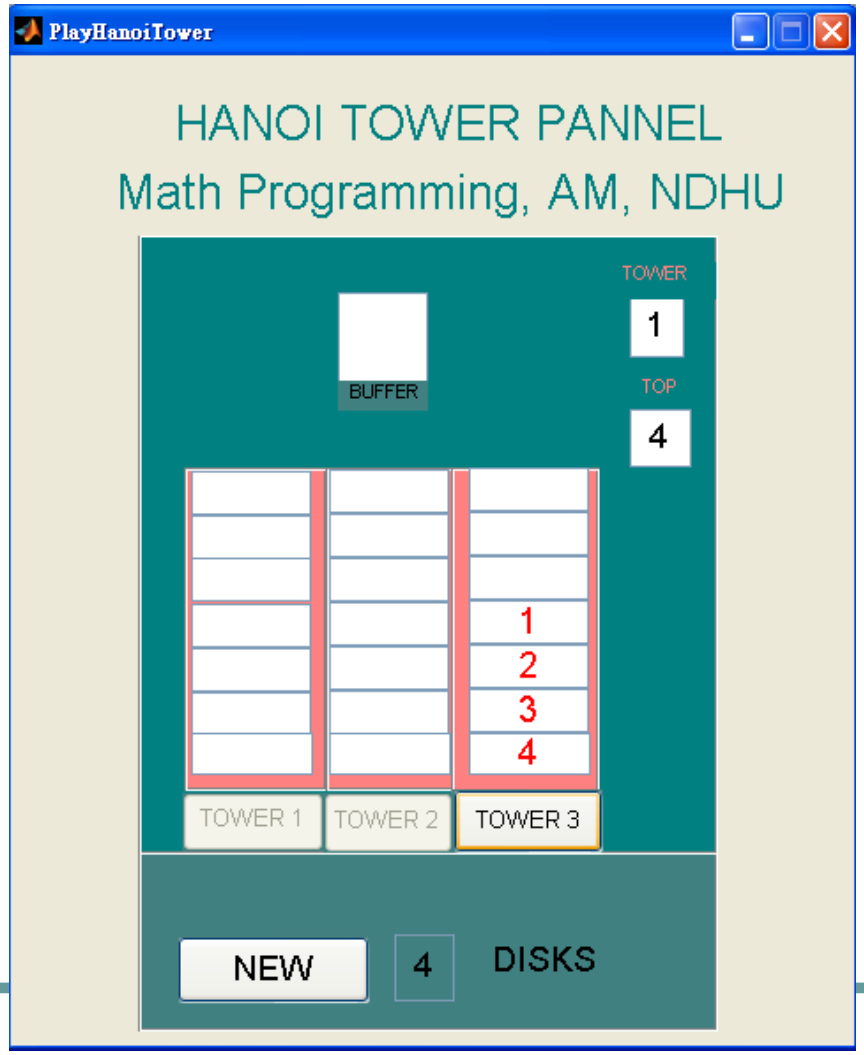
Move 3 disks from stack 1 to 2



Move disk 4

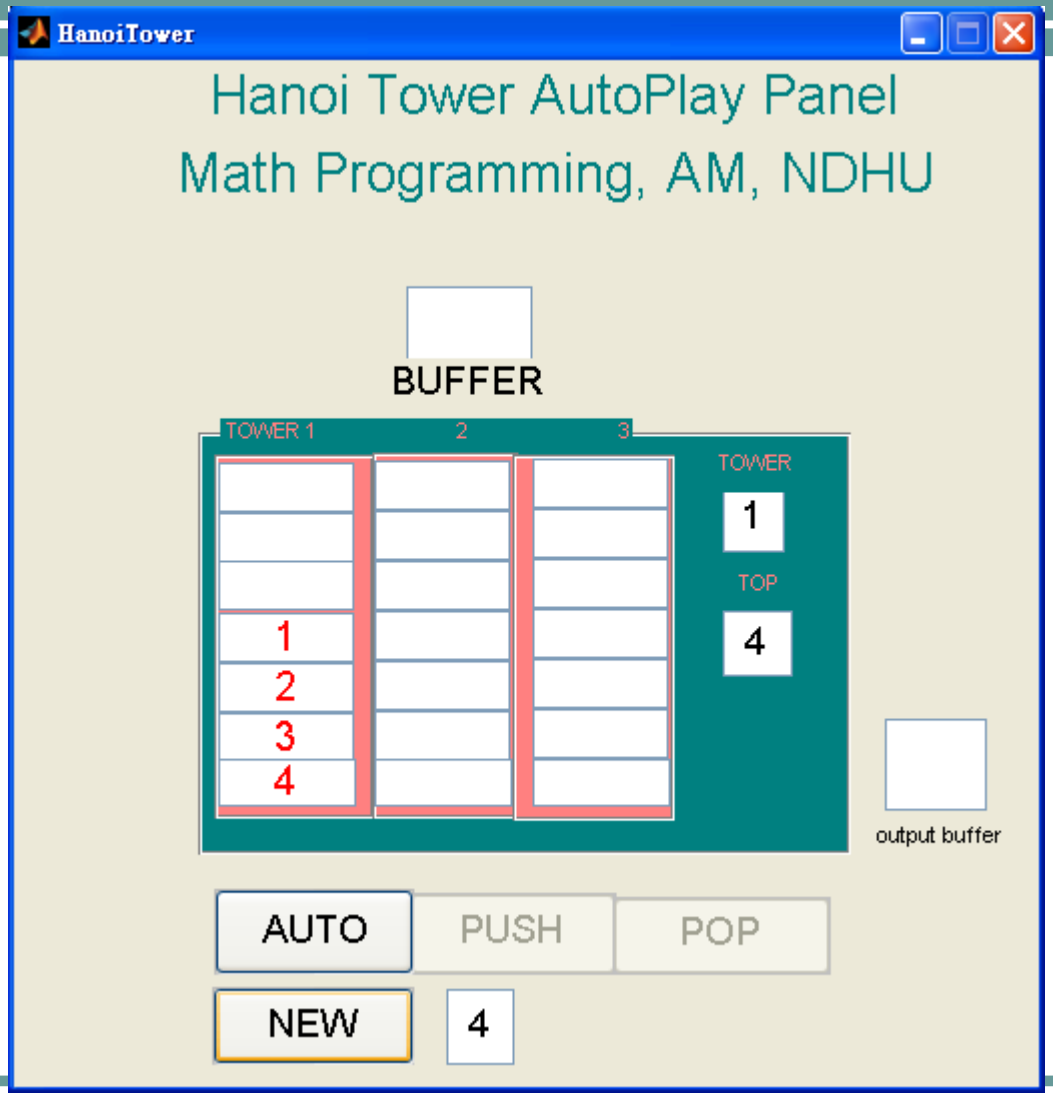

PlayHanoiTower
HANOI TOWER PANNEL
Math Programming, AM, NDHU
TOWER
1
BUFFER
TOP
1
1
2
3
4
TOWER 1
TOWER 2
TOWER 3
NEW
4
DISKS

Move 3 disks from stack 2 to 3



Hanoi-tower auto-play panel

HanoiTower.m
HanoiTower.fig


HanoiTower
Hanoi Tower AutoPlay Panel
Math Programming, AM, NDHU
BUFFER
TOWER 1
2
3
TOWER
1
TOP
4
1
2
3
4
output buffer
AUTO
PUSH
POP
NEW
4