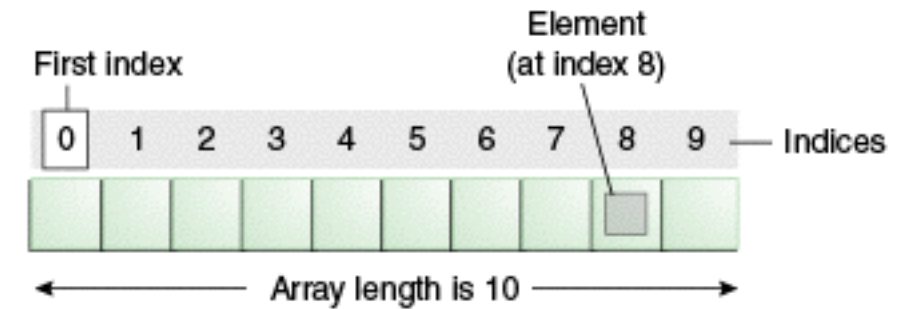


# Converting from infix to prefix

- Need a simple algorithm to convert any infix expression to corresponding prefix one
- $\underline{(2 + (3 * 4))}$
- $+ 2 * 3 4$
- $(+ 2 \underline{(* 3 4)})$
- Respects the order of evaluation

# Stacks

- *Data Structure*
  - Way of organising data in computer
  - Example: Arrays
- Operations
  - Add item to the data
  - Look at item
  - Remove item



## Arrays

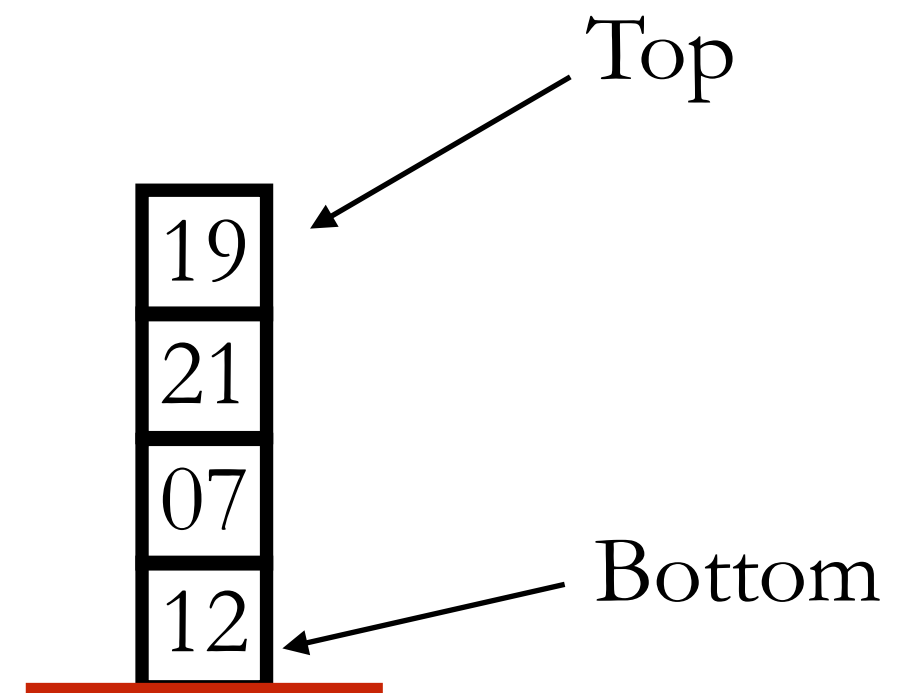
`i[0]:=3;`

`x:=i[1]+i[2];`

*Fixed length*

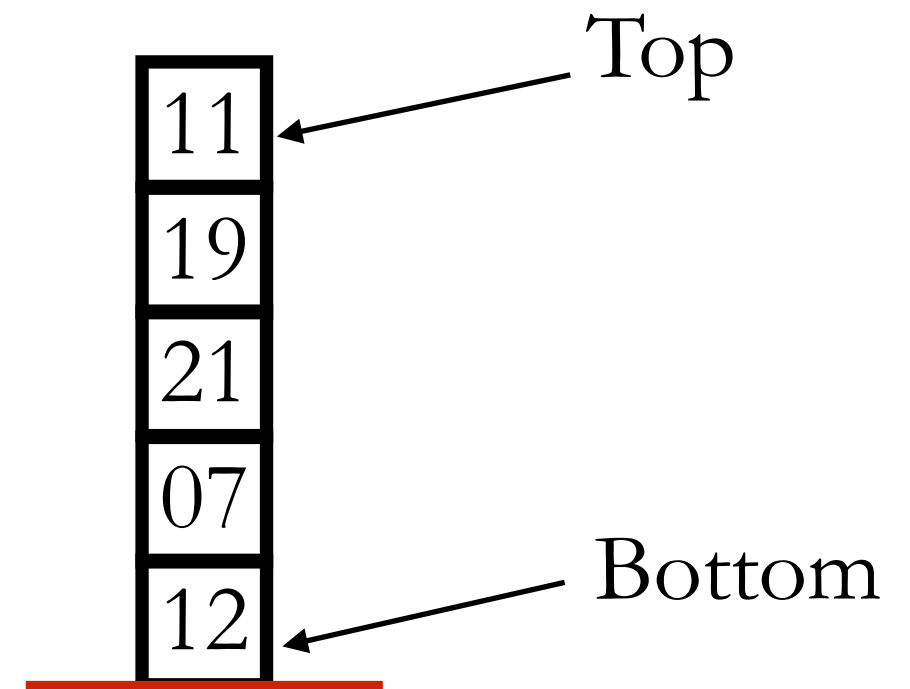
# Stacks

- *Push* item on
- PUSH 11



# Stacks

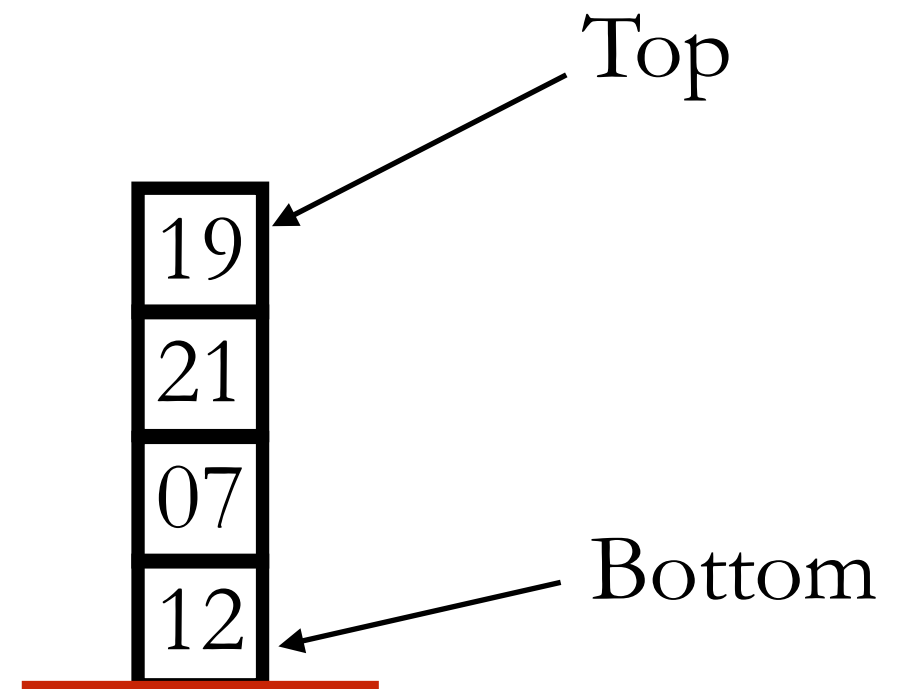
- *Push* item on
  - PUSH 11
- *Pop* item off
  - POP



# Stacks

- *Push* item on
  - PUSH 11
- *Pop* item off
  - POP
- PUSH (O)
- POP (X)

11



# Examples

- Input string:
  - HELLO

Stack H E L L O

O/P O L L E H

Operations O O O O O X X X X X

# Examples

- Input string:
  - HELLO
- Create:
  - HELLO

Stack **H**

O/P **HELLO**

**Operations** O X O X O X O X O X

# Examples

- Input string:
  - HELLO
- Create:
  - OHLEL

Operations O OOOOOX

Stack H E L L O

O/P O

**Not possible**

Stacks are fast and simple

Somewhat restrictive

*Dynamic* data structure



# Back to infix to prefix conversion

1. Reverse the expression
2. Read expression one character at a time:

**The Stack Method**

- “)”: Push onto stack
- Operator: Push onto stack
- Operand: Push on and pop off (straight to output)
- “(“: Keep popping stack until “)” is encountered

3. Reverse the output

# Example

- Input string:

- $(3 + 1)$

- Reverse:

- ) 1 + 3 (

Stack    )   + 3

O/P    13+

Reverse to get + 3 1

Operations    OOXOOXXX

# Example

1. Reverse the expression
2. Read expression one character at a time:
  - “)”: Push onto stack
  - Operator: Push onto stack
  - Operand: Push on and pop off (straight to output)
  - “(“: Keep popping stack until “)” is encountered
3. Reverse the output