CS 240

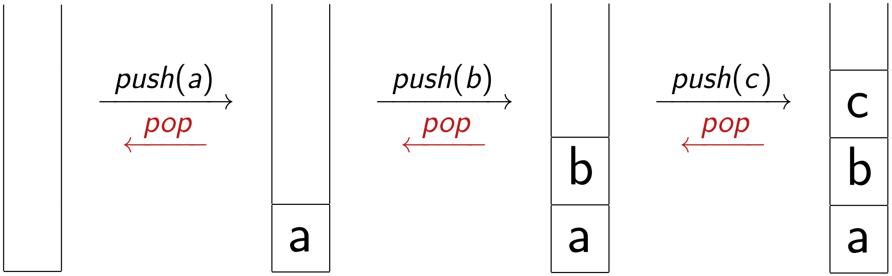
Data Structures and Algorithms I

Alex Vondrak

ajvondrak@csupomona.edu

October 12, 2011

Stacks



- `isEmpty()`
- `size()`
- `top()`

Stacks
Abstract Data Type

```
interface Stack {
   public void push(int item);
   public int pop()
      throws StackUnderflowException;
   public int top()
      throws StackUnderflowException;
   public boolean isEmpty();
   public int size();
}

class SomeStackImplementation implements Stack {
   /* must implement all the methods */
}
```

Stack Algorithm: Balanced Parentheses

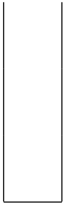
```
String parens = "(()())";
Stack s = new Stack(); // of chars

for(int i = 0; i < parens.length(); i++) {
   char c = parens.charAt(i);
   if (c == '(') s.push(c);
   else          s.pop();
}
if (s.isEmpty()) System.out.println("Balanced");
else             System.out.println("Unbalanced");
```

Stack-Based Evaluation

- If we see a number, push it to the data stack
- If we see an operator, pop the operands and push the result

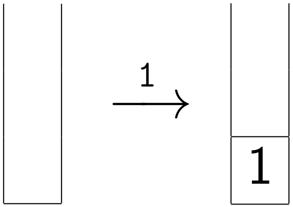
Example (`1 2 + 3 * 4 -`)



Stack-Based Evaluation

- If we see a number, push it to the data stack
- If we see an operator, pop the operands and push the result

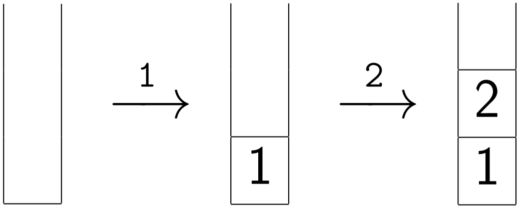
Example (1 2 + 3 * 4 -)



Stack-Based Evaluation

- If we see a number, push it to the data stack
- If we see an operator, pop the operands and push the result

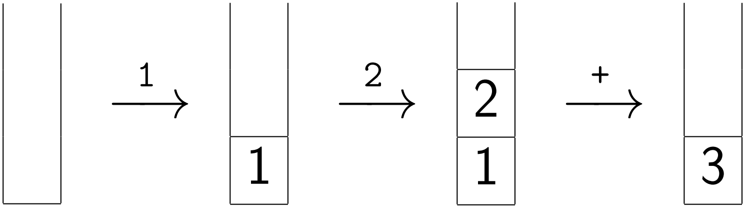
Example (1 2 + 3 * 4 -)



Stack-Based Evaluation

- If we see a number, push it to the data stack
- If we see an operator, pop the operands and push the result

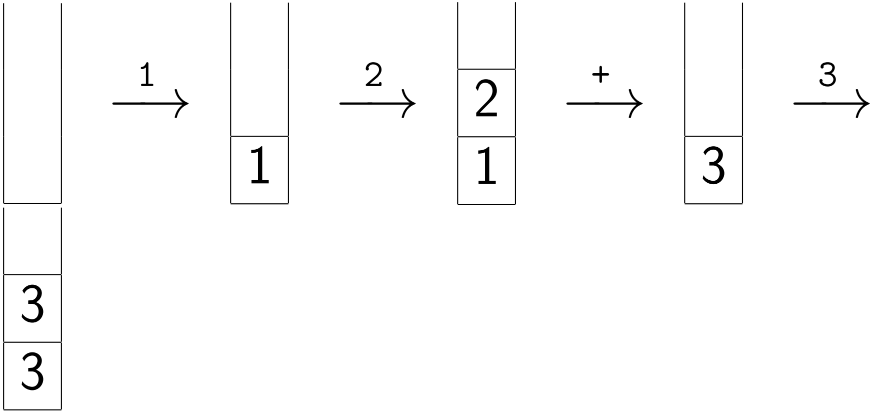
Example (`1 2 + 3 * 4 -`)



Stack-Based Evaluation

- If we see a number, push it to the data stack
- If we see an operator, pop the operands and push the result

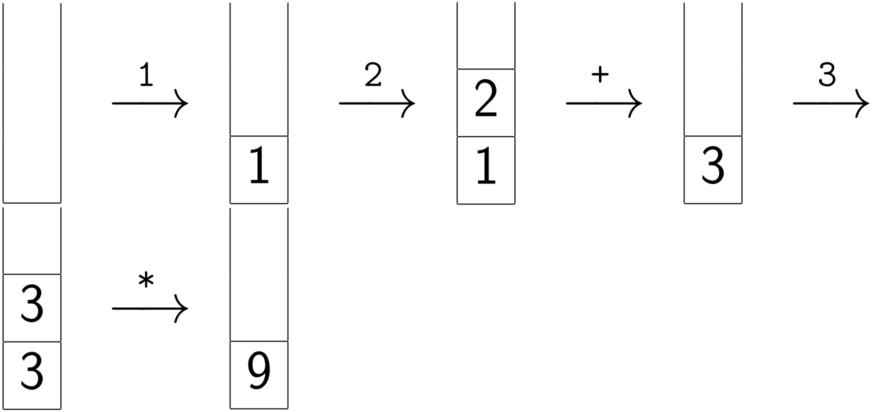
Example (1 2 + 3 * 4 -)



Stack-Based Evaluation

- If we see a number, push it to the data stack
- If we see an operator, pop the operands and push the result

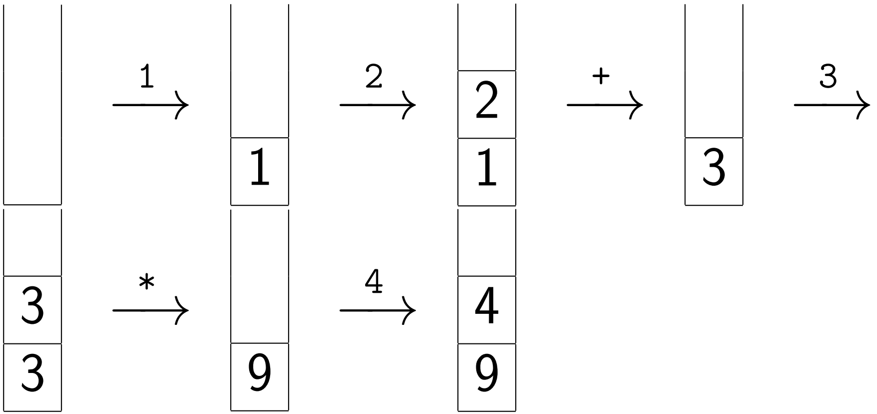
Example (1 2 + 3 * 4 -)



Stack-Based Evaluation

- If we see a number, push it to the data stack
- If we see an operator, pop the operands and push the result

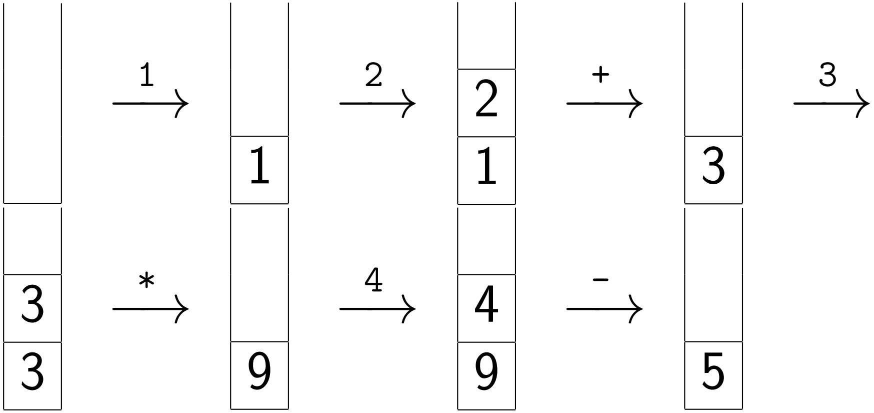
Example (`1 2 + 3 * 4 -`)



Stack-Based Evaluation

- If we see a number, push it to the data stack
- If we see an operator, pop the operands and push the result

Example (`1 2 + 3 * 4 -`)



Converting Infix To Postfix

- If you see a left parenthesis, push it onto the stack
- If you see a number, write it to the output
- If you see an operator, push it onto the stack
- Otherwise, next symbol should be a right parenthesis, and the top of the stack should be an operator
    - Pop the operator and write it to the output
    - Top of the stack should be a left parenthesis, so pop and discard
- At the end of the input, stack should be empty

Examples (Worked Out In Class)

- `((1 + 2) * 3)`
- `((1 + 2) * (3 + 4))`