PES UNIVERSITY

Data Structures

Dilip Maripuri
Associate Professor
Department of Computer Applications

Data Structures

Session Outline

PES UNIVERSITY

Stacks

Applications

Role of Data Structures in Managing Expressions

PES UNIVERSITY

- **Applying Stack Concepts to**
  - **Infix to Postfix Conversion**
  - Evaluation of Postfix Expression
  - Infix to Prefix Conversion
  - Prefix to Postfix Conversion

Infix to Postfix Conversion

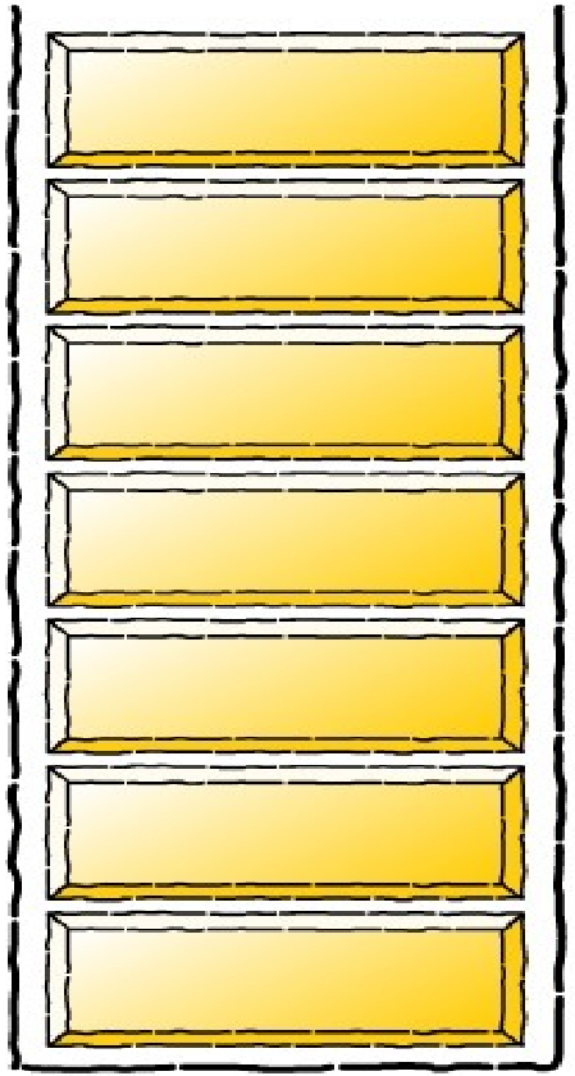
- Scan the Infix expression left to right
  - If the character x is an operand
    - Output the character into the Postfix Expression
  - If the character x is a left or right parenthesis
    - If the character is “(“
      - Push it into the stack
    - If the character is “)”
      - Repeatedly pop and output all the operators/characters until “(“ is popped from the stack.

Infix to Postfix Conversion

- Scan the Infix expression left to right
  - If the character x is a is a regular operator
    - Step 1: Check the character y currently at the top of the stack.
    - Step 2: If Stack is empty or y='(' or y is an operator of lower precedence than x, then push x into stack.
    - Step 3: If y is an operator of higher or equal precedence than x, then pop and output y and push x into the stack.
  - When all characters in infix expression are processed repeatedly pop the character(s) from the stack and output them until the stack is empty.

Data Structures

Infix to Postfix Conversion



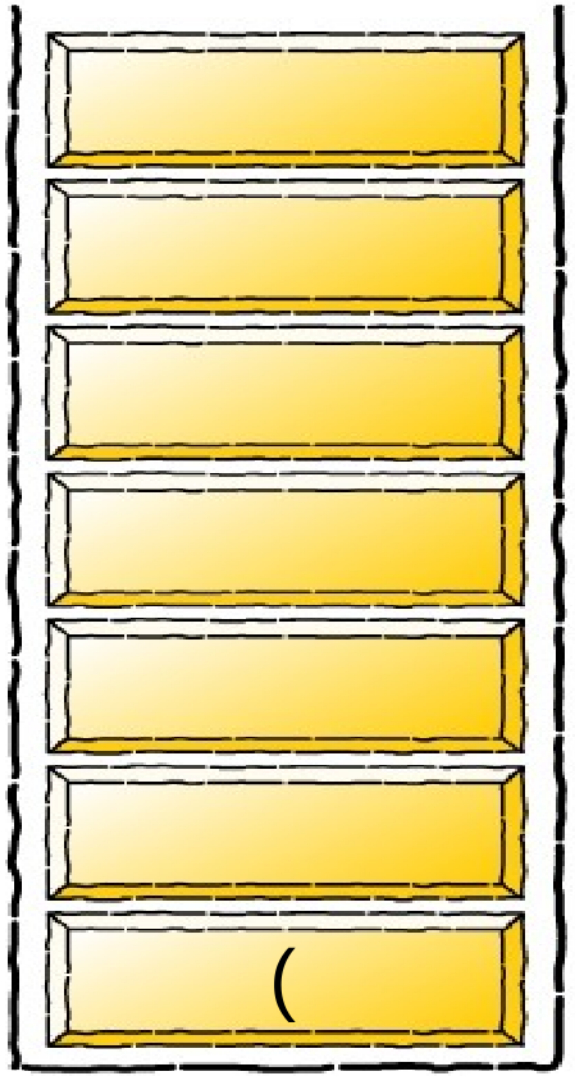
Infix Expression

(a + b - c) * d – (e + f)

Postfix Expression

Infix to Postfix Conversion

PES UNIVERSITY

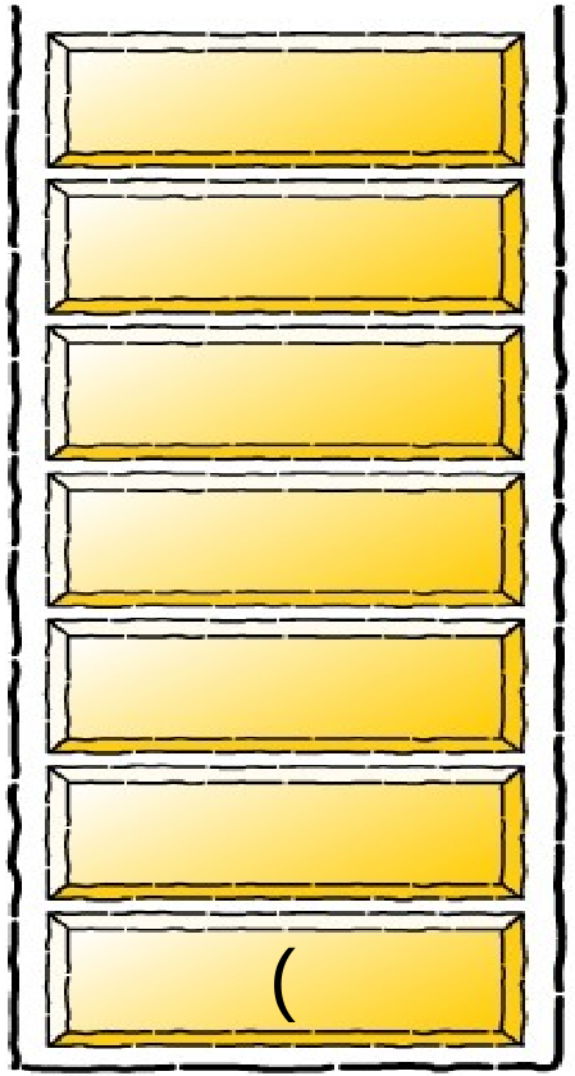


Infix Expression

a + b - c) * d – (e + f)

Postfix Expression

Infix to Postfix Conversion



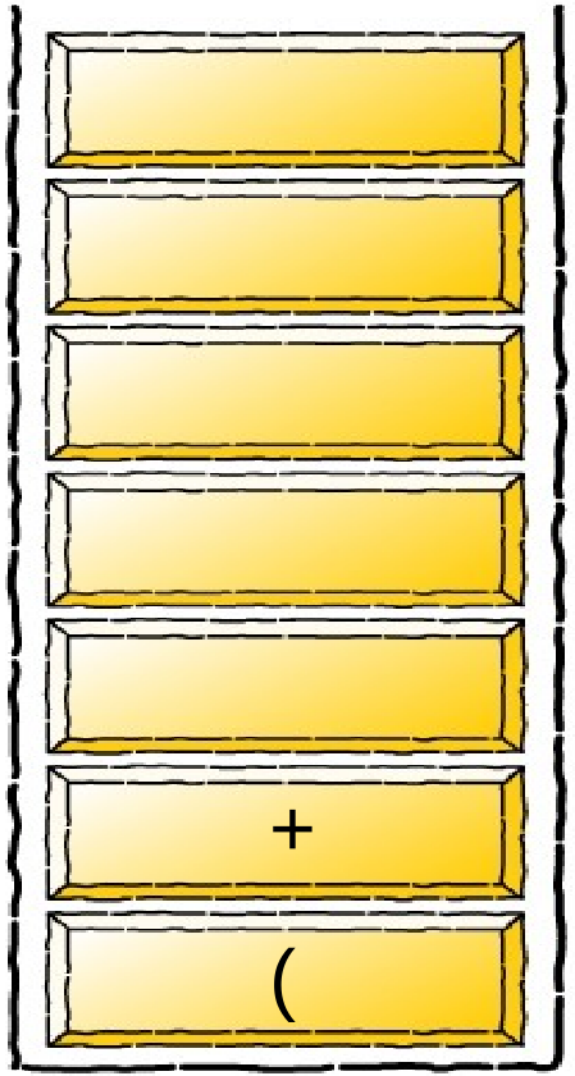
Infix Expression

+ b - c) * d – (e + f)

Postfix Expression

a

Infix to Postfix Conversion



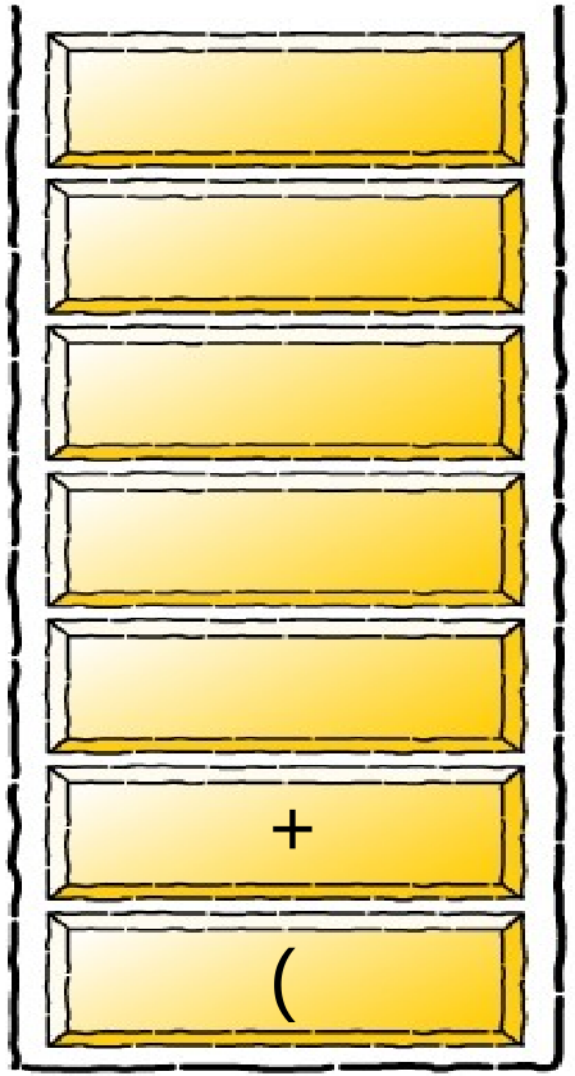
Infix Expression

b - c) * d – (e + f)

Postfix Expression

a

Infix to Postfix Conversion



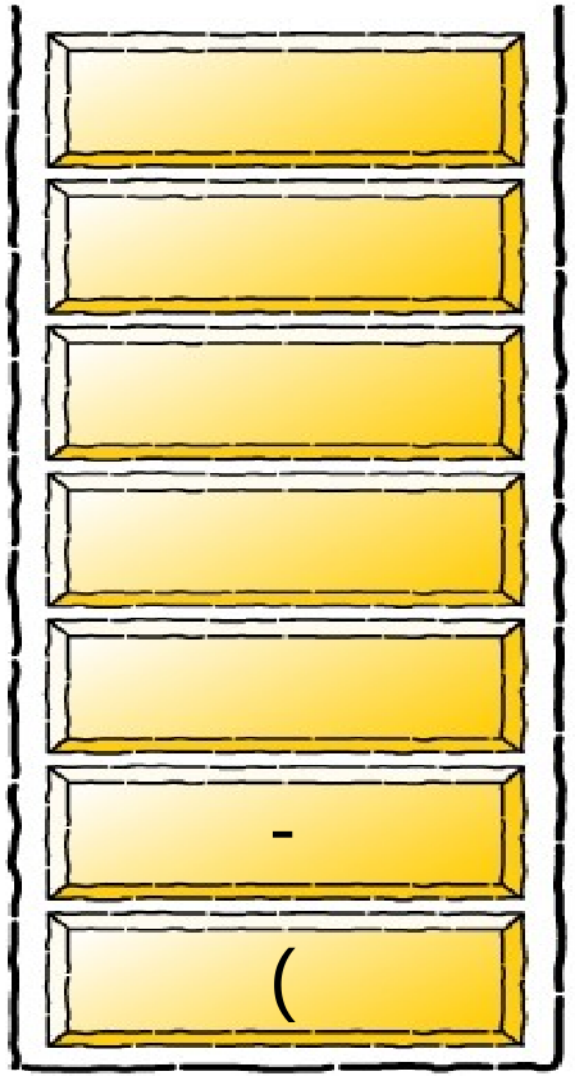
Infix Expression

- c) * d – (e + f)

Postfix Expression

a b

Infix to Postfix Conversion



Infix Expression

c) * d – (e + f)

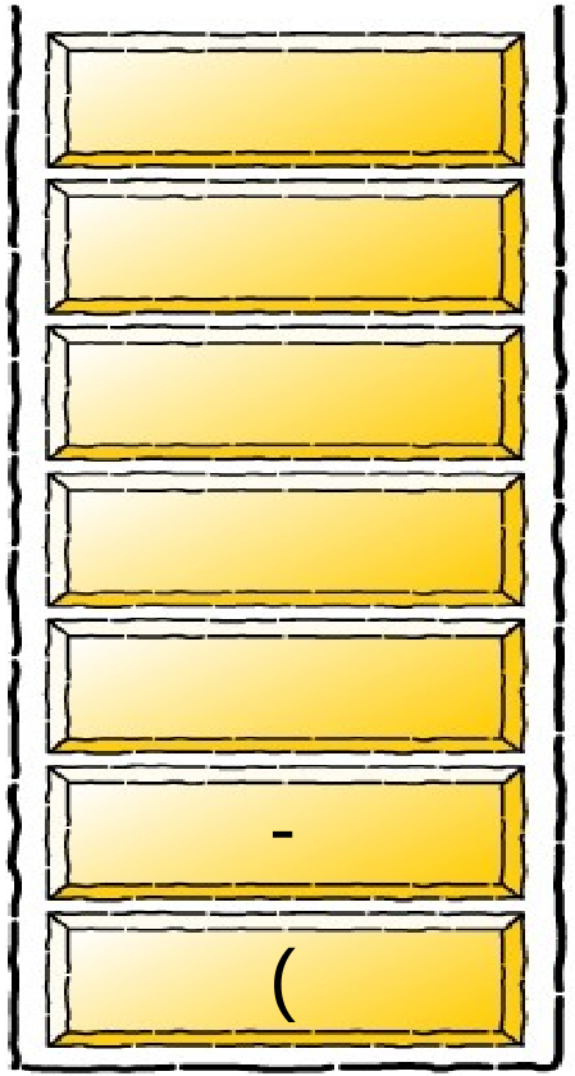
Postfix Expression

a b +

Data Structures

Infix to Postfix Conversion

PES UNIVERSITY



Infix Expression

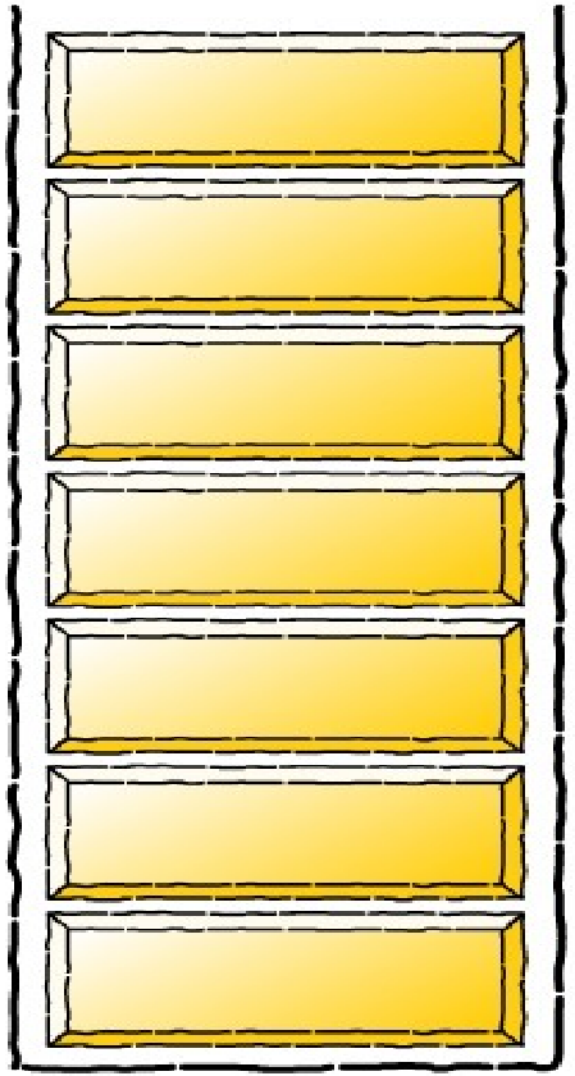
) * d – (e + f)

Postfix Expression

a b + c

Data Structures

Infix to Postfix Conversion



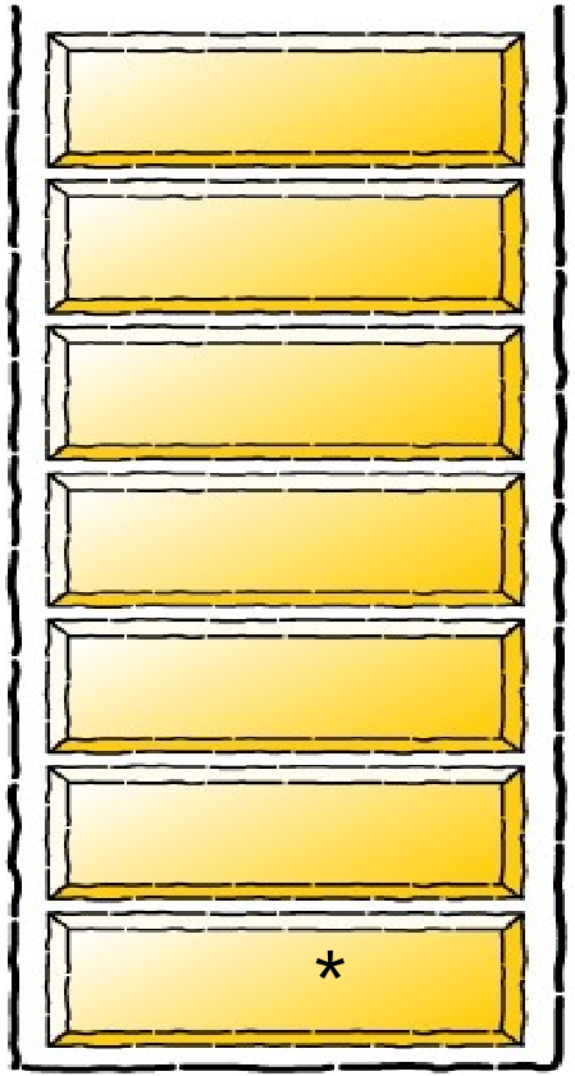
Infix Expression

```
* d – ( e + f )
```

Postfix Expression

```
a b + c -
```

Infix to Postfix Conversion



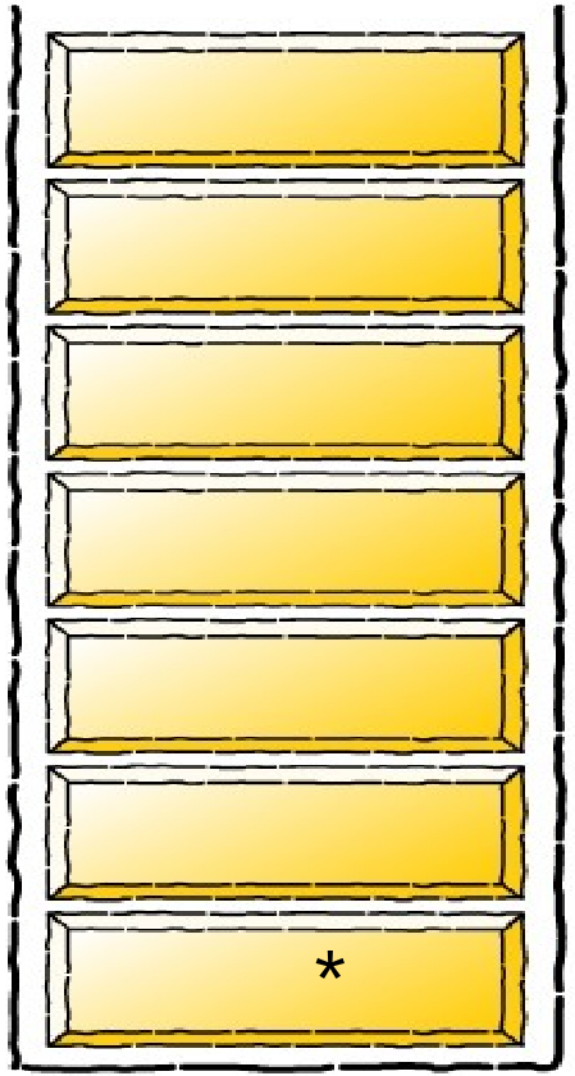
Infix Expression

d – (e + f)

Postfix Expression

a b + c -

Infix to Postfix Conversion



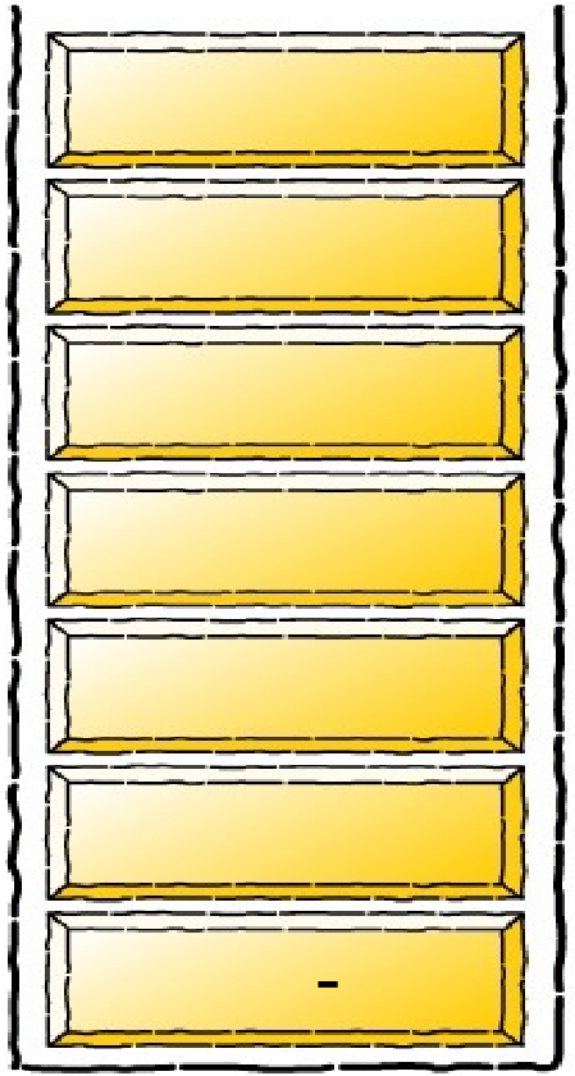
Infix Expression

– (e + f)

Postfix Expression

a b + c - d

Infix to Postfix Conversion



Infix Expression

(e + f)

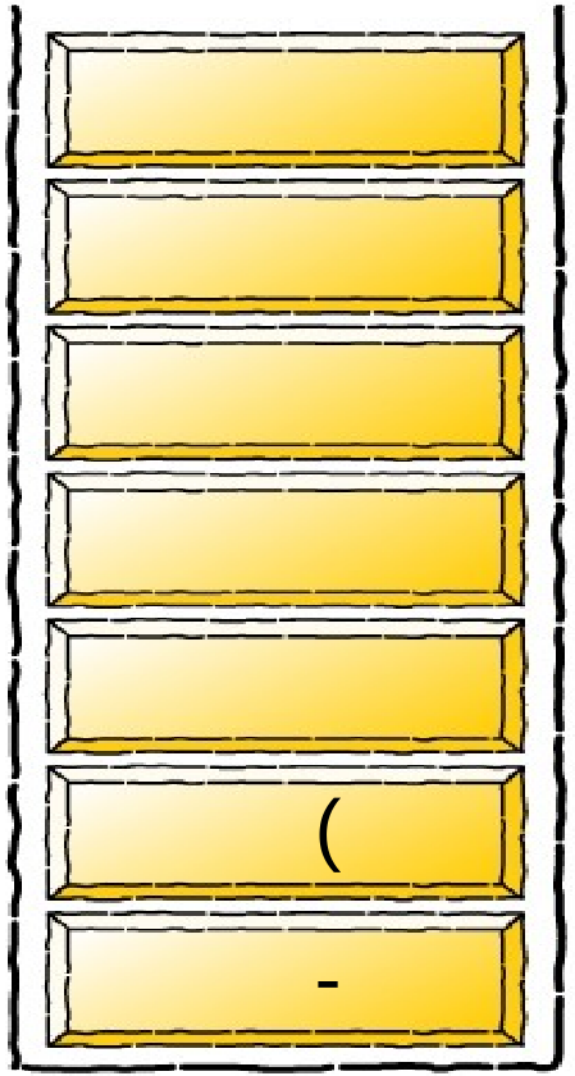
Postfix Expression

a b + c – d *

Data Structures

Infix to Postfix Conversion

PES UNIVERSITY



Infix Expression

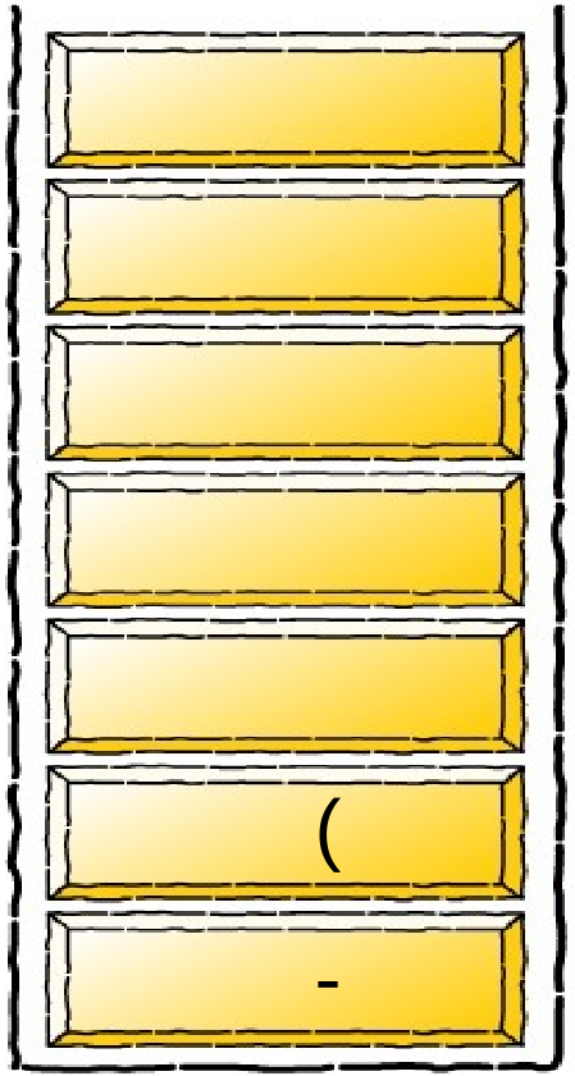
e + f)

Postfix Expression

a b + c – d *

Data Structures

Infix to Postfix Conversion



Infix Expression

+ f)

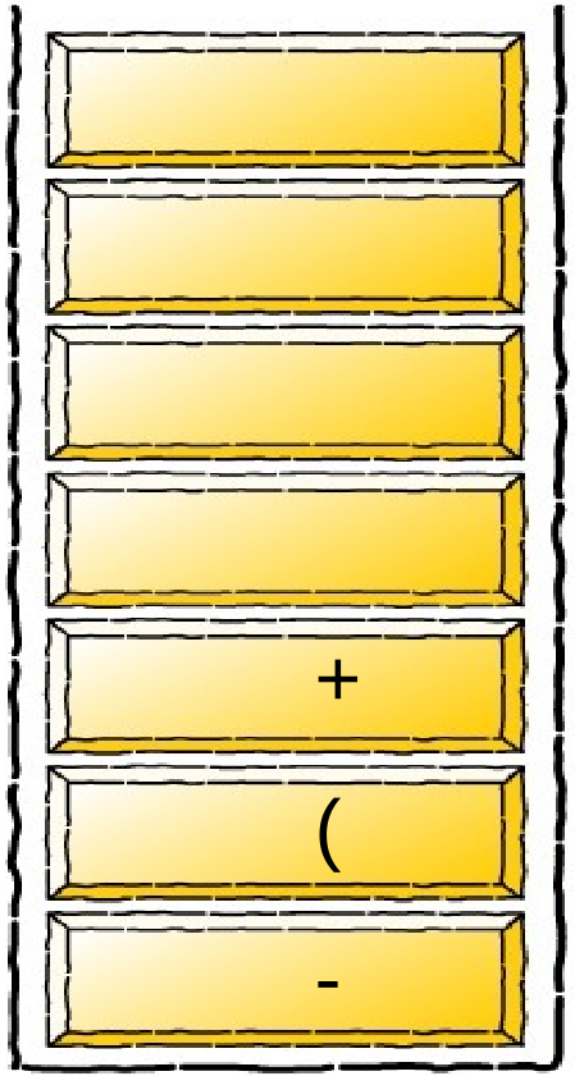
Postfix Expression

a b + c – d * e

Data Structures

Infix to Postfix Conversion

PES UNIVERSITY



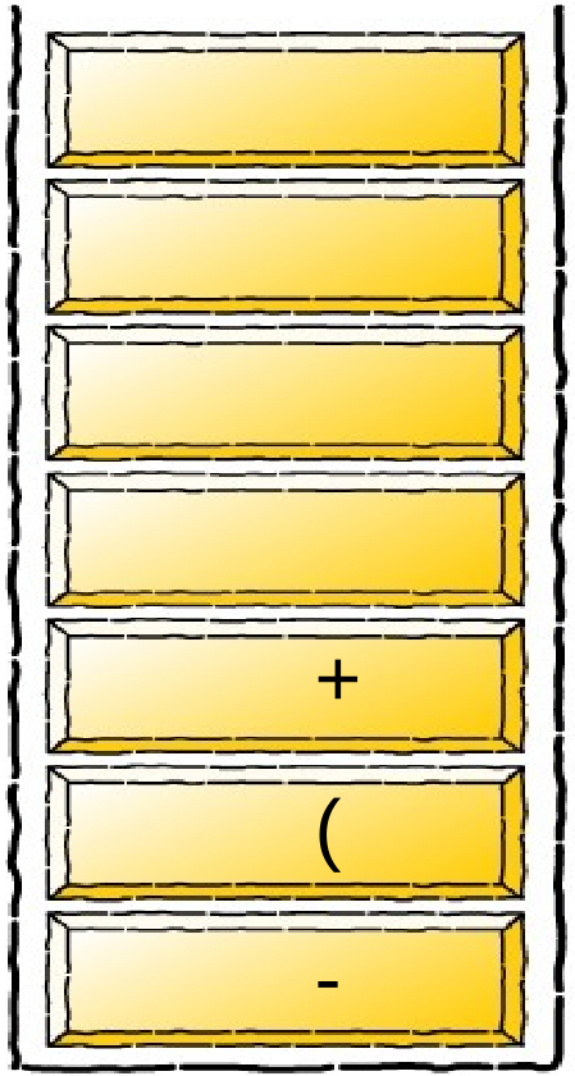
Infix Expression

f)

Postfix Expression

a b + c – d * e

Infix to Postfix Conversion



Infix Expression

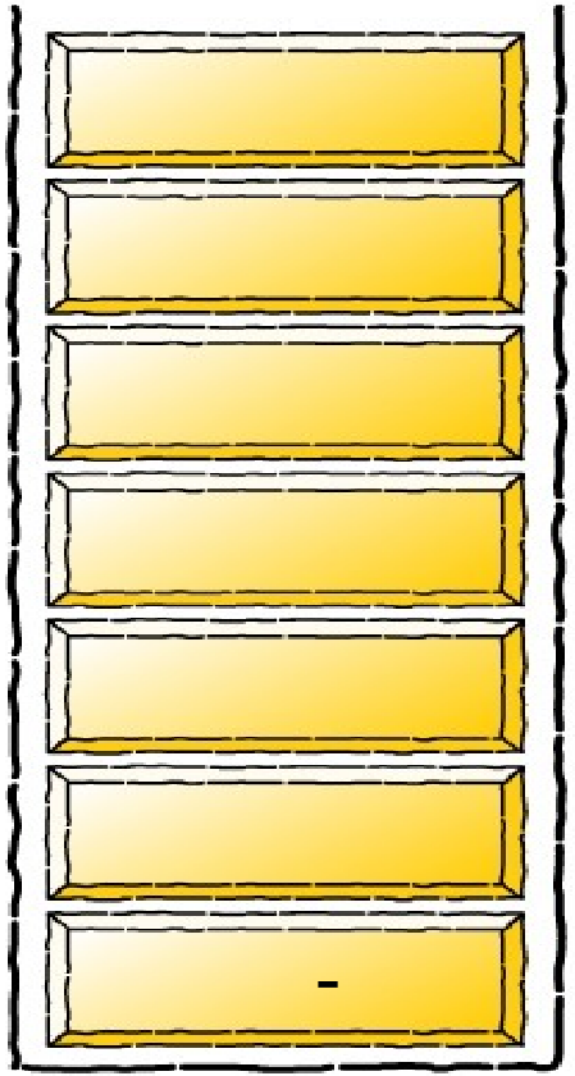
)

Postfix Expression

a b + c – d * e f

Infix to Postfix Conversion

PES UNIVERSITY



Infix Expression

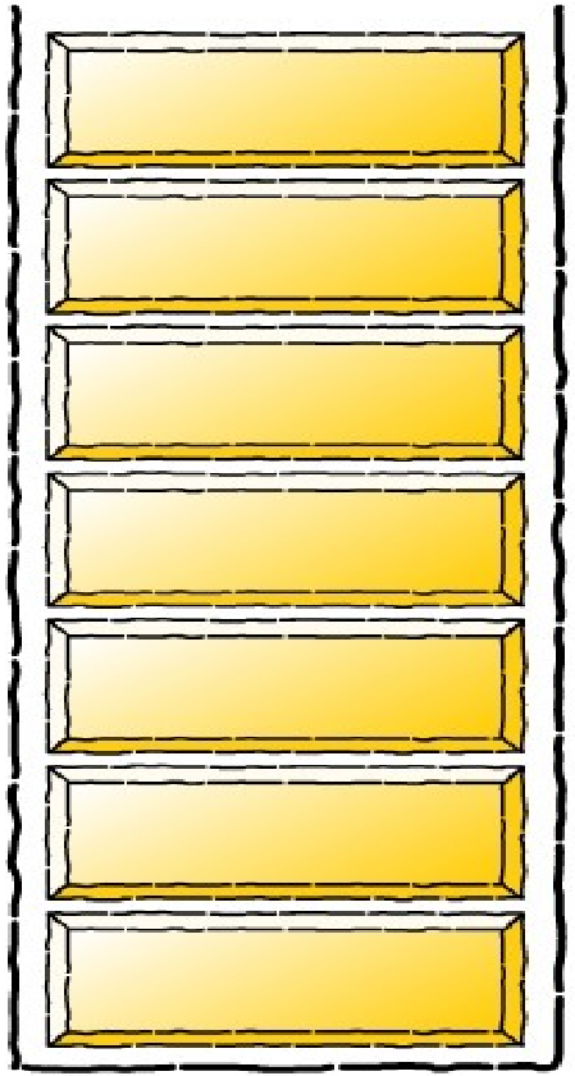
Postfix Expression

a b + c – d * e f +

Data Structures

Infix to Postfix Conversion

PES UNIVERSITY



Infix Expression

Postfix Expression

a b + c – d * e f + -

Infix to Postfix Conversion

Step	Infix Expression	Stack	Postfix Expression	Action
1	A * (B + C) / D			Read 'A'
2	* (B + C) / D		A	Append 'A'
3	(B + C) / D	*	A	Push '*'
4	B + C) / D	*(	A	Push '('
5	+ C) / D	*(	AB	Append 'B'
6	C) / D	*(+	AB	Push '+'
7	) / D	*(+	ABC	Append 'C'
8	/ D	*(	AB+C	Pop '+' and Append, then Pop '('
9	D	/	AB+C*	Pop '*' and Append, Push '/'
10		/	AB+C*D	Append 'D'
11			AB+C*D/	Pop '/' and Append

Infix to Postfix Conversion


```
maripuri@maripuri:~/Desktop/MCA SEM 1 - DS/DSC$ ./a.out

Program to Convert Infix to Postfix Expression

Enter the Infix Expression : A*(B+C)/F

        Scanned Char            Stack           Postfix Exp
        A                       #               A
        *                       #*              A
        (                       #*(             A
        B                       #*(             AB
        +                       #*(+            AB
        C                       #*(+            ABC
        )                       #*              ABC+
        /                       #/              ABC+*
        F                       #/              ABC+*F
Resultant Postfix Expression is : ABC+*F/
```

Infix to Postfix Conversion

PES UNIVERSITY

```
Enter the Infix Expression : (a+b)*(c/d)+e

        Scanned Char            Stack           Postfix Exp
        (                       #(
        a                       #(              a
        +                       #(+             a
        b                       #(+             ab
        )                       #               ab+
        *                       #*              ab+
        (                       #*(             ab+
        c                       #*(             ab+c
        /                       #*(/            ab+c
        d                       #*(/            ab+cd
        )                       #*              ab+cd/
        +                       #+              ab+cd/*
        e                       #+              ab+cd/*e
Resultant Postfix Expression is : ab+cd/*e+
```

Infix to Postfix Conversion

PES UNIVERSITY

```
Enter the Infix Expression : (((a+b)+c)*(d-e))/((f*g)+h)+i

        Scanned Char            Stack           Postfix Exp
        (                       #(
        (                       #((
        (                       #(((
        a                       #(((            a
        +                       #(((+           a
        b                       #(((+           ab
        )                       #((             ab+
        +                       #((+            ab+
        c                       #((+            ab+c
        )                       #(              ab+c+
        *                       #(*             ab+c+
        (                       #(*(            ab+c+
        d                       #(*(            ab+c+d
        -                       #(*(-           ab+c+d
```

Infix to Postfix Conversion

PES UNIVERSITY

-	#(*(-	ab+c+d
e	#(*(-	ab+c+de
)	#(*	ab+c+de-
)	#	ab+c+de-*
/	#/	ab+c+de-*
(	#/(	ab+c+de-*
(	#/((	ab+c+de-*
f	#/((	ab+c+de-*f
*	#/((*	ab+c+de-*f
g	#/((*	ab+c+de-*fg
)	#/(	ab+c+de-*fg*
+	#/(+	ab+c+de-*fg*
h	#/(+	ab+c+de-*fg*h
)	#/	ab+c+de-*fg*h+
+	#+	ab+c+de-*fg*h+/
i	#+	ab+c+de-*fg*h+/i

Resultant Postfix Expression is : ab+c+de-*fg*h+/i+

Role of Data Structures in Managing Expressions

PES UNIVERSITY

- **Try !!!**

$$-((a^b)*(c+d))/(e-f)+((g*h)/(i^j))$$

Role of Data Structures in Managing Expressions

PES UNIVERSITY

```
char s[MAX];
int Top = 0;
void push(char elem)            {       s[++Top] = elem;        }
char pop()                      {       return (s[Top--]);      }
int precd(char elem)
{       switch(elem)
        {       case '+' :
                case '-' : return 1;
                case '*' :
                case '/' : return 2;
                case '^' :
                case '$' : return 3;
                case '(' :
                case '#' : return 0;    }
}
```

```
#include<stdio.h>
#include<ctype.h>
#include<stdlib.h>
# include <string.h>
#define MAX 20
```

Role of Data Structures in Managing Expressions

PES UNIVERSITY

```
void main(void)
{
        char prefix[MAX],infix[MAX],postfix[MAX],ch,elem;
        int i = 0, j = 0, k=0;
        printf("\nEnter the Infix Expression : ");
        scanf("%s",infix);
        push('#');
```

Role of Data Structures in Managing Expressions

```
for(i=0;i<strlen(infix);i++)
{
        ch = infix[i];
        if(isalnum(ch))          postfix[j++] = ch;
        else
                if(ch == '(')
                        push(ch);
                else
                if(ch == ')')
                        while(s[Top] != '(')
                                postfix[j++] = pop();
                        elem = pop();
                else
                {
                        while(precd(s[Top]) >= precd(ch))
                                postfix[j++] = pop();
                        push(ch);
                }
}
```

```
while(s[Top] != '#')
        postfix[j++] = pop();
postfix[j] = '\0';
printf("%s",postfix);
```

Role of Data Structures in Managing Expressions

PES UNIVERSITY

- **Applying Stack Concepts to**
  - Infix to Postfix Conversion
  - **Infix to Prefix Conversion**
  - Prefix to Postfix Conversion
  - Evaluation of Postfix Expression

Infix to Prefix Conversion

PES UNIVERSITY

- Reverse the Infix Expression:
  - Reverse the given infix expression.
  - Replace every '(' with ')' and vice versa in the reversed expression.
- Convert the resultant expression into postfix form by applying the infix to postfix conversion strategy
- Reverse the resultant postfix expression to get the prefix expression

Infix to Prefix Conversion

PES UNIVERSITY

```
Enter the Infix Expression : A*(B+C)/F

Reversed Infix Expression is : F/(C+B)*A
        Scanned Char            Stack           Postfix Exp
        F                       #               F
        /                       #/              F
        (                       #/(             F
        C                       #/(             FC
        +                       #/(+            FC
        B                       #/(+            FCB
        )                       #/              FCB+
        *                       #/*             FCB+
        A                       #/*             FCB+A

Resultant Nearly Postfix Expression is : FCB+A*/
Resultant Prefix Expression is : /*A+BCF
```

Infix to Prefix Conversion


```
Enter the Infix Expression : (A+B)*(C/D)+E

Reversed Infix Expression is : E+(D/C)*(B+A)
        Scanned Char              Stack              Postfix Exp
        E                         #                  E
        +                         #+                 E
        (                         #+(                E
        D                         #+(                ED
        /                         #+(/               ED
        C                         #+(/               EDC
        )                         #+                 EDC/
        *                         #+*                EDC/
        (                         #+*(               EDC/
        B                         #+*(               EDC/B
        +                         #+*(+              EDC/B
        A                         #+*(+              EDC/BA
        )                         #+*                EDC/BA+

Resultant Nearly Postfix Expression is : EDC/BA+*+
Resultant Prefix Expression is : +*+AB/CDE
```

Infix to Prefix Conversion

PES UNIVERSITY

```
Enter the Infix Expression : 2+3*4

Reversed Infix Expression is : 4*3+2
        Scanned Char              Stack              Postfix Exp
        4                         #                  4
        *                         #*                 4
        3                         #*                 43
        +                         #+                 43*
        2                         #+                 43*2

Resultant Nearly Postfix Expression is : 43*2+
Resultant Prefix Expression is : +2*34
```

Infix to Prefix Conversion

PES UNIVERSITY

```
Enter the Infix Expression : p*(q+r)/s

Reversed Infix Expression is : s/(r+q)*p
        Scanned Char            Stack           Postfix Exp
        s                       #               s
        /                       #/              s
        (                       #/(             s
        r                       #/(             sr
        +                       #/(+            sr
        q                       #/(+            srq
        )                       #/              srq+
        *                       #/*             srq+
        p                       #/*             srq+p

Resultant Nearly Postfix Expression is : srq+p*/
Resultant Prefix Expression is : /*p+qrs
```

Infix to Prefix Conversion

PES UNIVERSITY

```
Enter the Infix Expression : (n*m+o)/p

Reversed Infix Expression is : p/(o+m*n)
        Scanned Char            Stack           Postfix Exp
        p                       #               p
        /                       #/              p
        (                       #/(             p
        o                       #/(             po
        +                       #/(+            po
        m                       #/(+            pom
        *                       #/(+*           pom
        n                       #/(+*           pomn
        )                       #/              pomn*+

Resultant Nearly Postfix Expression is : pomn*+/
Resultant Prefix Expression is : /+*nmop
```

Role of Data Structures in Managing Expressions

PES UNIVERSITY

- **Applying Stack Concepts to**
  - Infix to Postfix Conversion
  - Infix to Prefix Conversion
  - **Prefix to Postfix Conversion**
  - Evaluation of Postfix Expression

Prefix to Postfix Conversion

PES UNIVERSITY

- **Scan the Prefix Expression from Right to Left**
  - Initialize an empty stack.
  - Start scanning the prefix expression from the right end.
- **Process Each Character**
  - If the character x is an operand
    - Push x onto the stack.

Prefix to Postfix Conversion

- **Process Each Character**
  - If the character x is an operator
    - Pop two operands from the stack. (operand1 and operand2).
    - Create a string by concatenating operand1, operand2, and x in this order. This string is a partial postfix expression.
    - Push this string back to the stack.
- **Complete Processing:**
  - Continue the process until the end of the prefix expression is reached.

Prefix to Postfix Conversion

- **Final Output**
  - At this point, the stack will contain only one element.
  - This element is the required postfix expression.

Role of Data Structures in Managing Expressions

PES UNIVERSITY

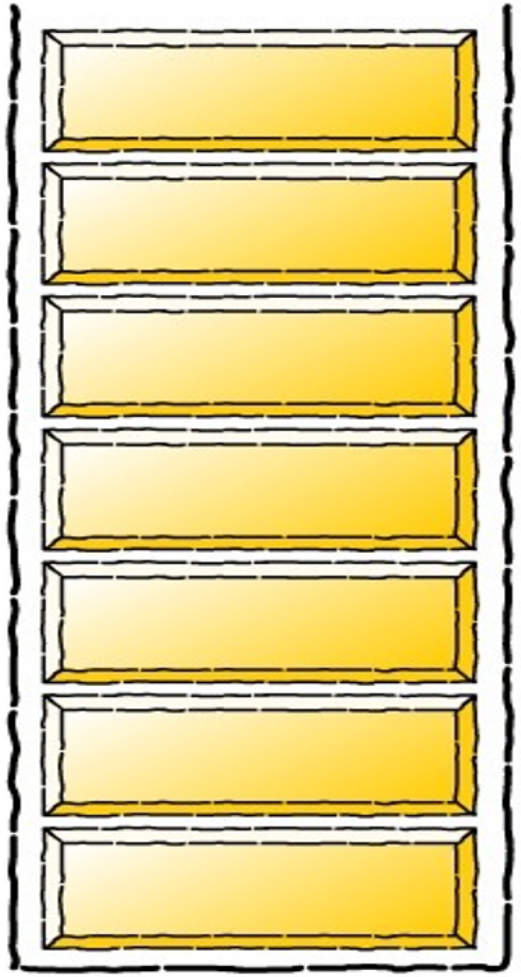
- **Applying Stack Concepts to**
  - Infix to Postfix Conversion
  - Infix to Prefix Conversion
  - Prefix to Postfix Conversion
  - **Evaluation of Postfix Expression**

Evaluation of Postfix Expression

- Scan the postfix expression left to right
  - If the character x is a digit
    - Push it into the stack
  - If the *character* is an operator (+,-,*,/)
    - pop two values from the stack
    - Perform the operation
    - Push the result of the operation into the stack
- When all characters in postfix expression are processed pop the value from the stack and output it.

Evaluation of Postfix Expression

PES UNIVERSITY

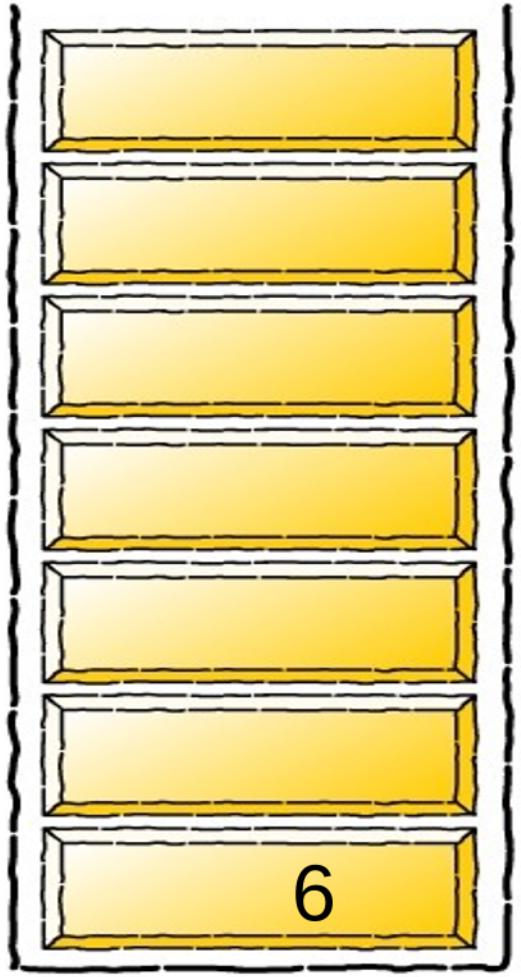


Postfix Expression

6 5 2 3 + 8 * + 3 + *

€ € € = €

Evaluation of Postfix Expression

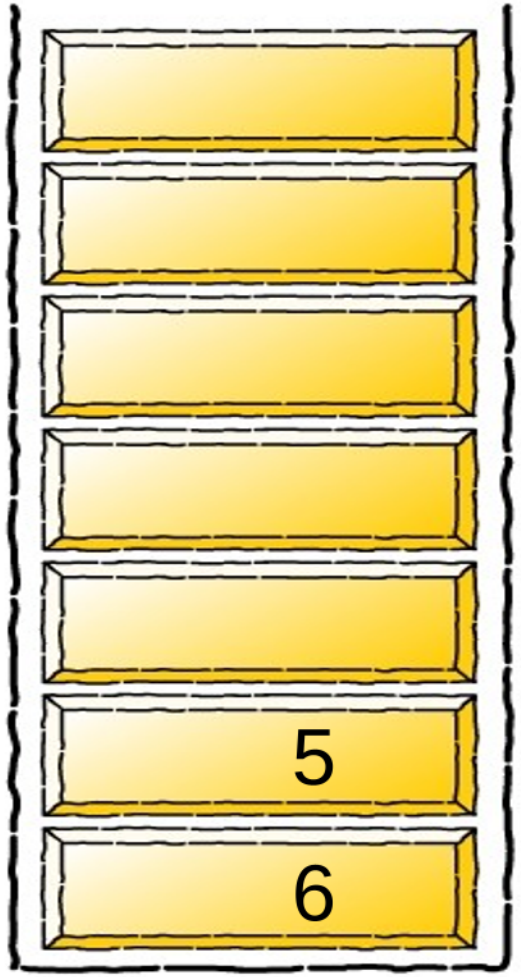


Postfix Expression

5 2 3 + 8 * + 3 + *

€ € € = €

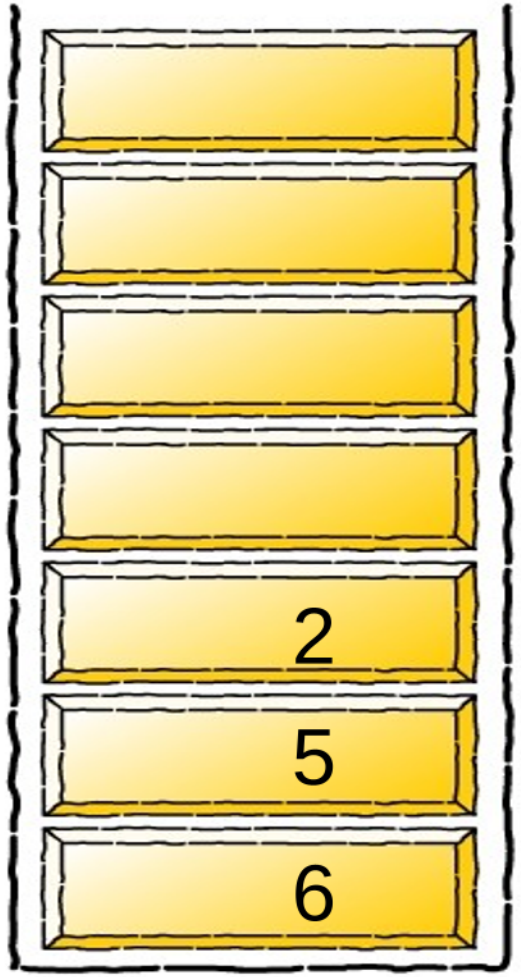
Evaluation of Postfix Expression



Postfix Expression

2 3 + 8 * + 3 + *

Evaluation of Postfix Expression



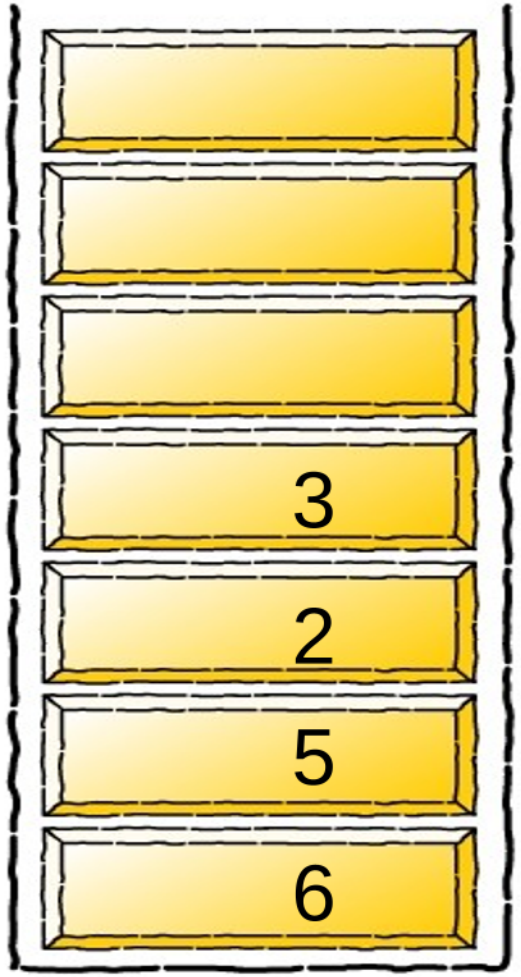
Postfix Expression

3 + 8 * + 3 + *

€ € € = €

Evaluation of Postfix Expression

PES UNIVERSITY

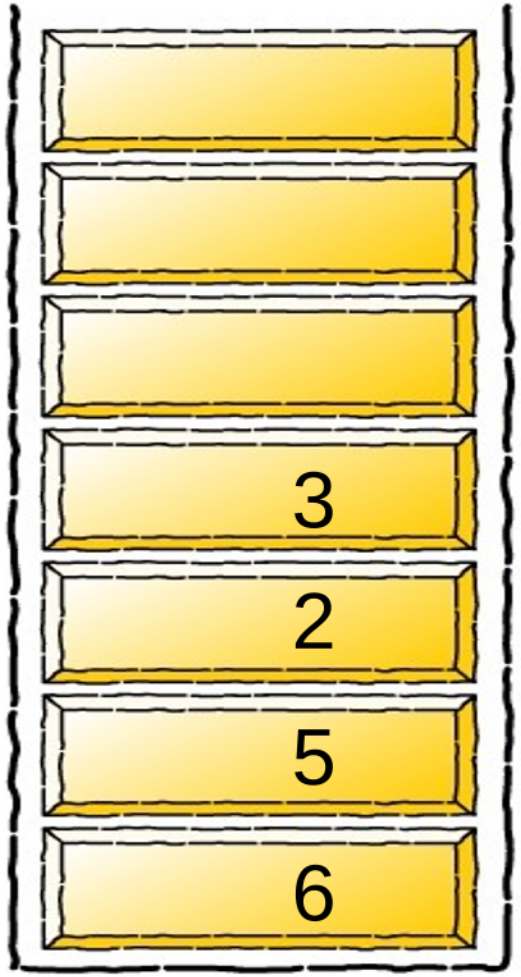


Postfix Expression

+ 8 * + 3 + *

€ € € = €

Evaluation of Postfix Expression

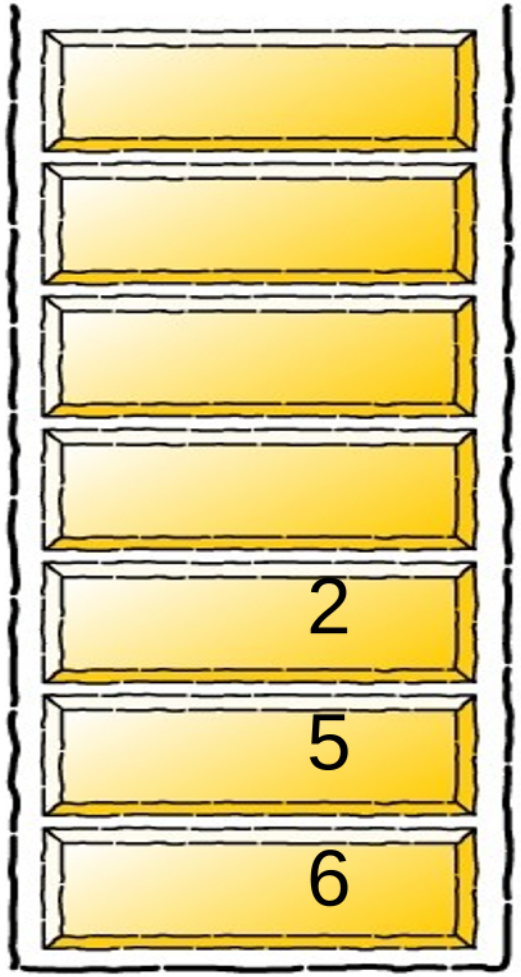


Postfix Expression

8 * + 3 + *

€ + € = €

Evaluation of Postfix Expression

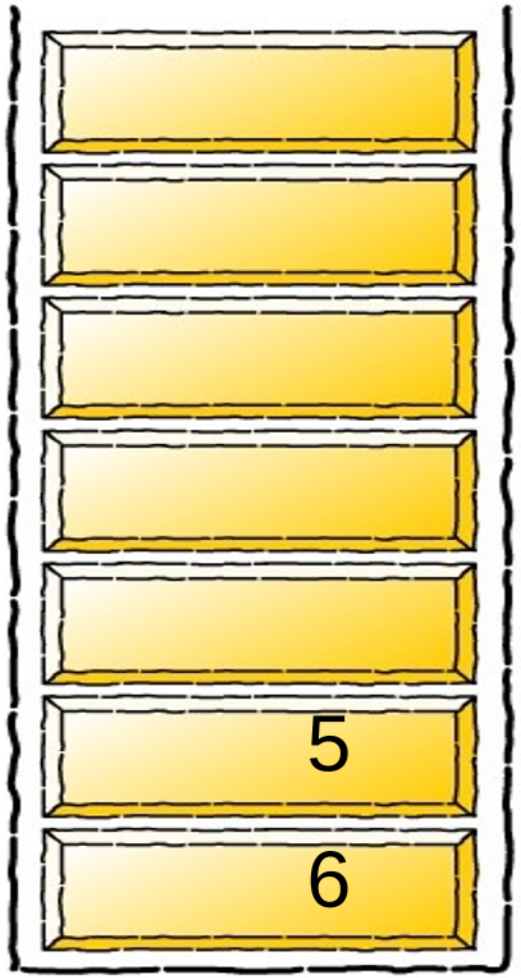


Postfix Expression

8 * + 3 + *

€ + 33= €

Evaluation of Postfix Expression



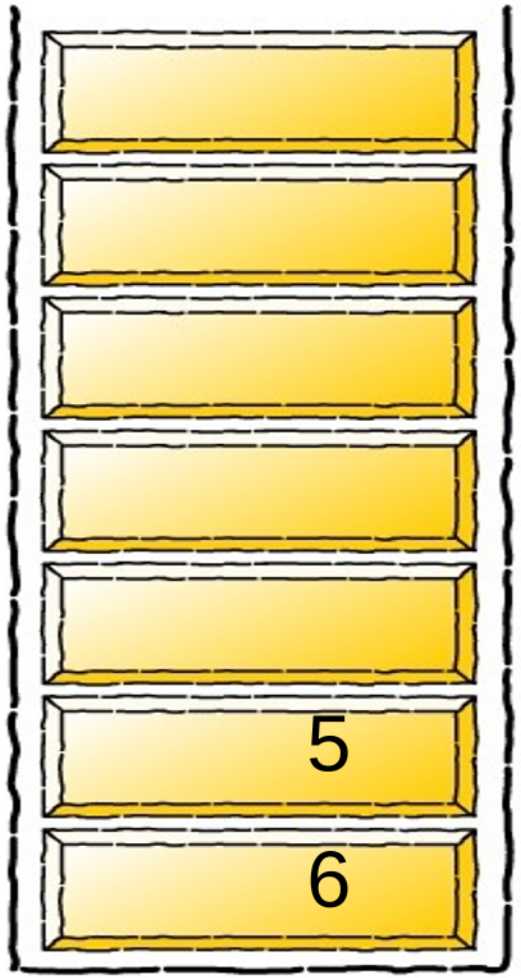
Postfix Expression

8 * + 3 + *

2 + 3 = €

Evaluation of Postfix Expression

PES UNIVERSITY

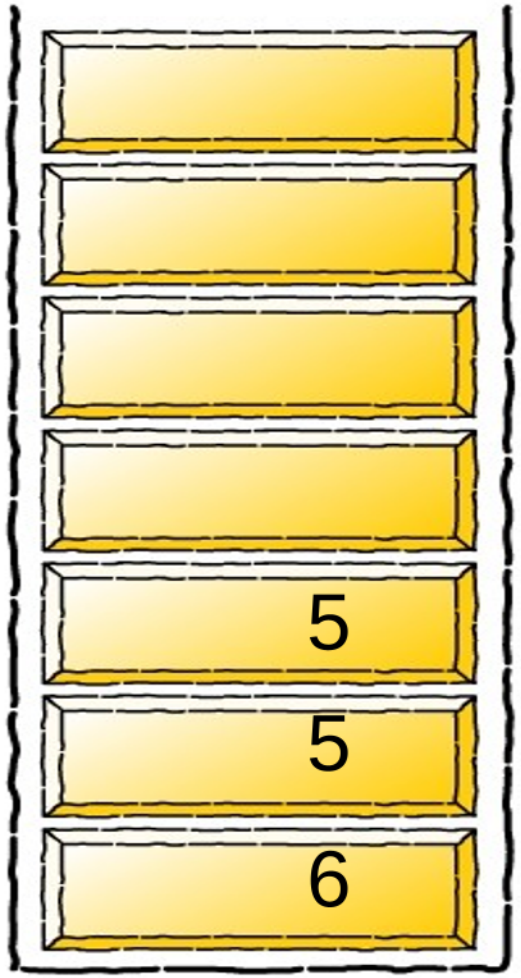


Postfix Expression

8 * + 3 + *

2 + 3 = 5

Evaluation of Postfix Expression

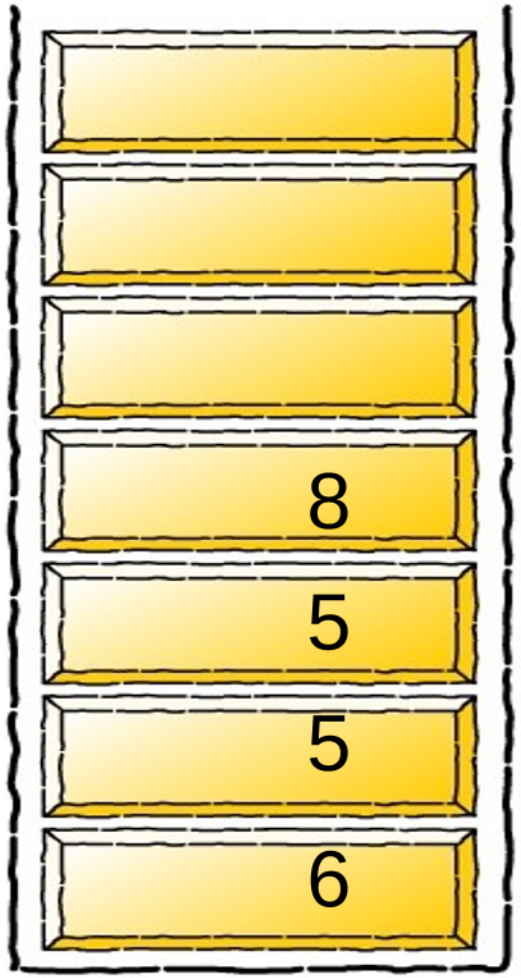


Postfix Expression

8 * + 3 + *

€ € € = €

Evaluation of Postfix Expression

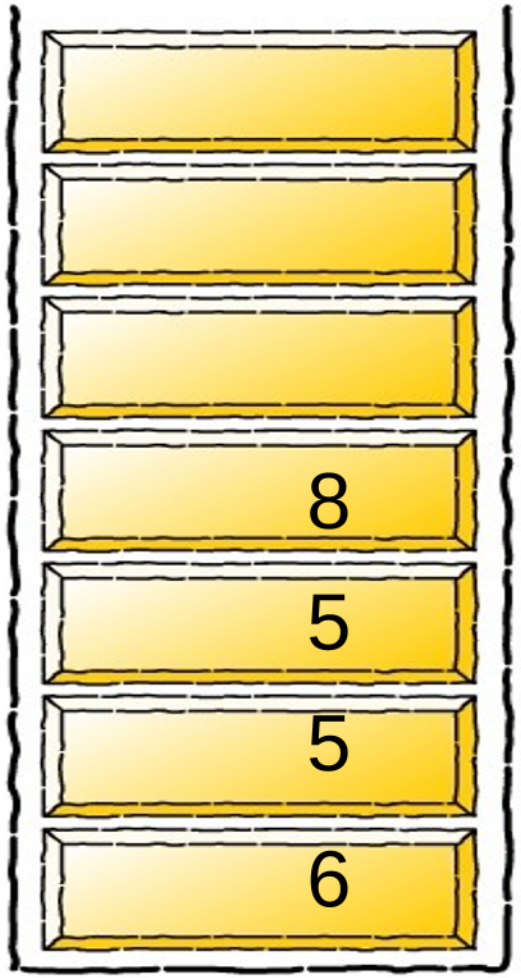


Postfix Expression

*** + 3 + ***

€ € € = €

Evaluation of Postfix Expression



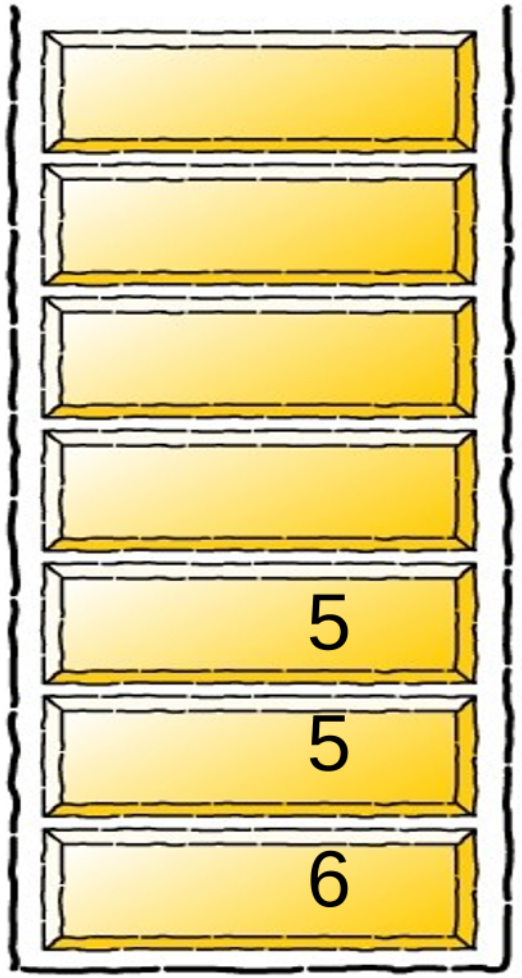
Postfix Expression

+ 3 + *

€ * € = €

Evaluation of Postfix Expression

PES UNIVERSITY

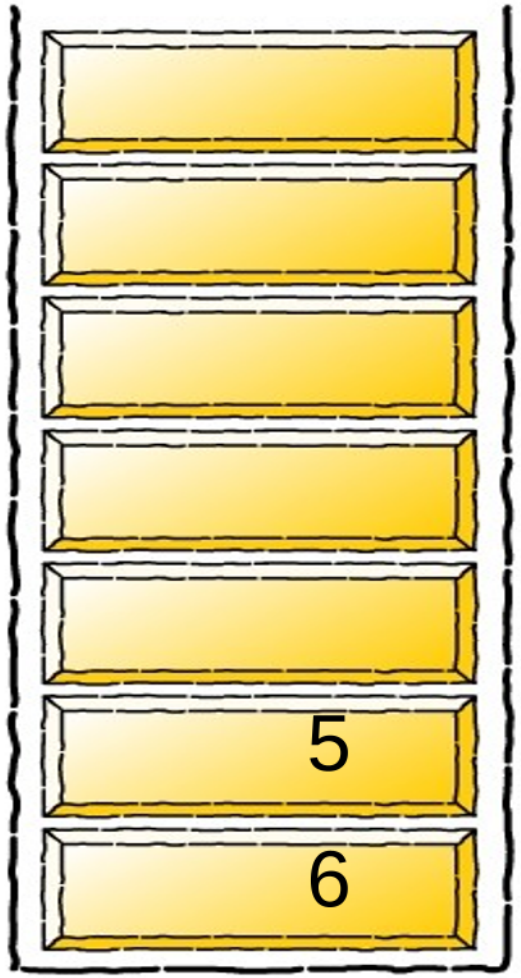


Postfix Expression

+ 3 + *

€ * 8 = €

Evaluation of Postfix Expression

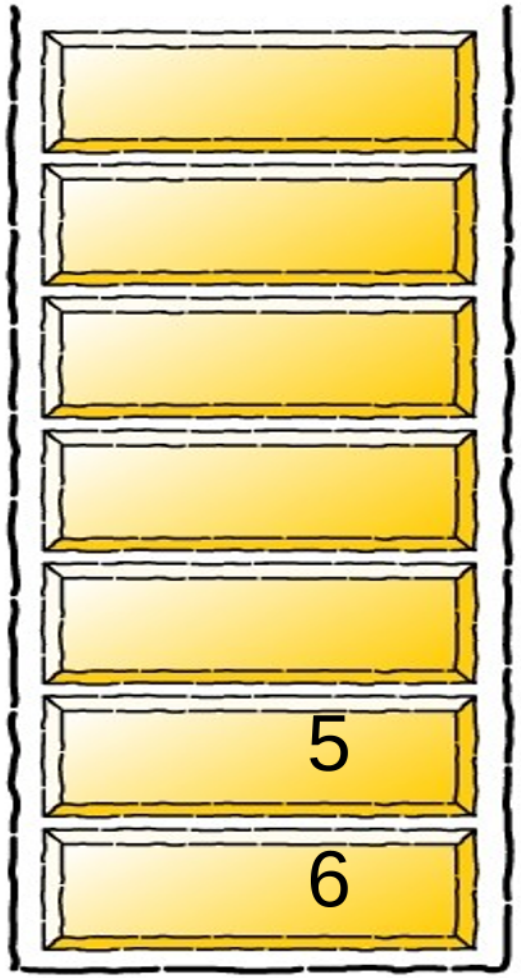


Postfix Expression

+ 3 + *

5 * 8 = €

Evaluation of Postfix Expression



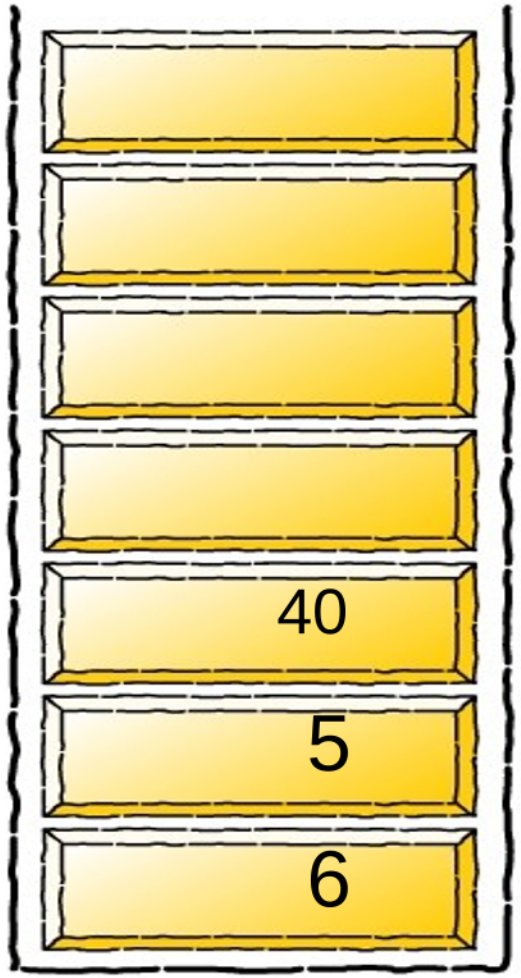
Postfix Expression

+ 3 + *

5 * 8 = 40

Evaluation of Postfix Expression

PES UNIVERSITY

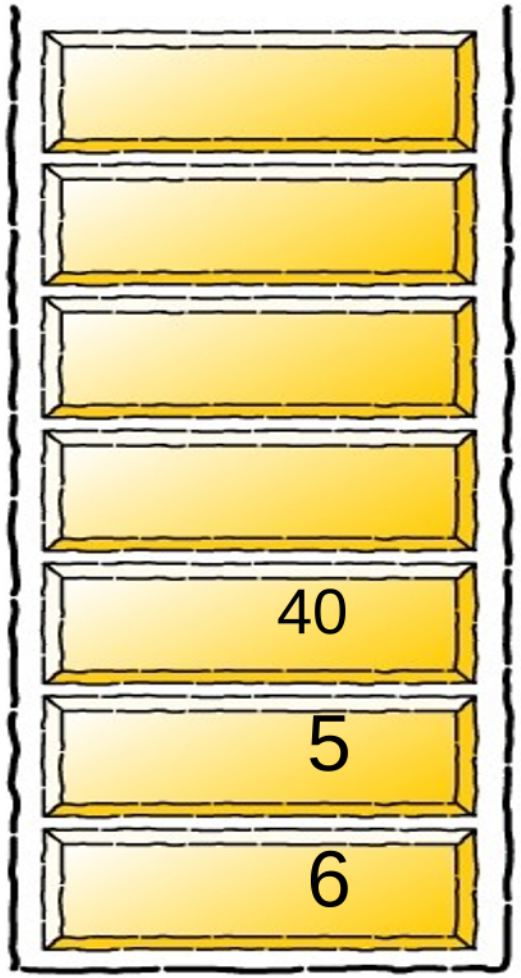


Postfix Expression

+ 3 + *

€ € € = €

Evaluation of Postfix Expression



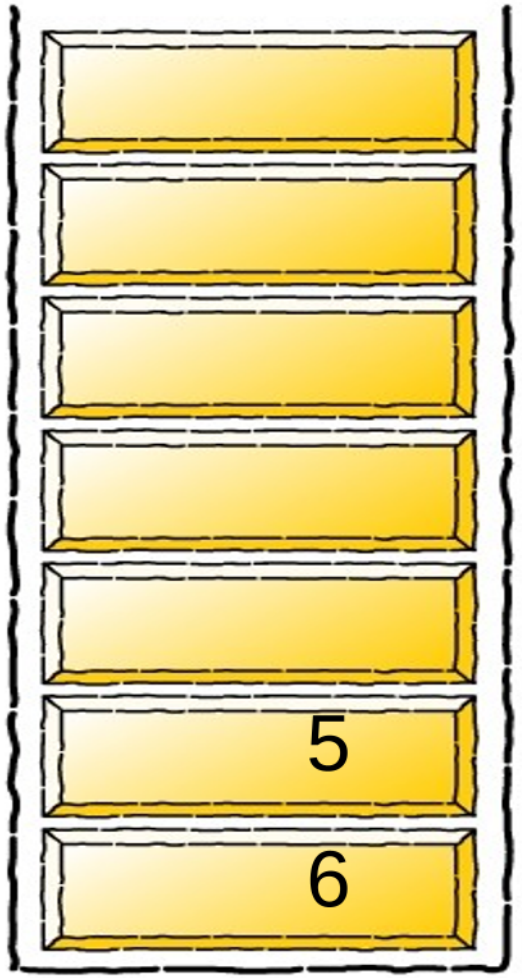
Postfix Expression

3 + *

€ + € = €

Evaluation of Postfix Expression

PES UNIVERSITY

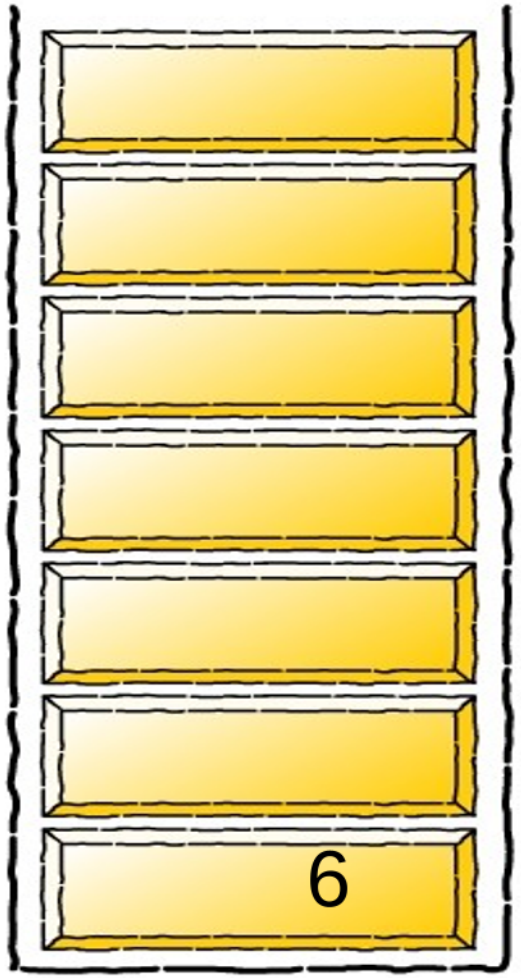


Postfix Expression

3 + *

€ + 40 = €

Evaluation of Postfix Expression

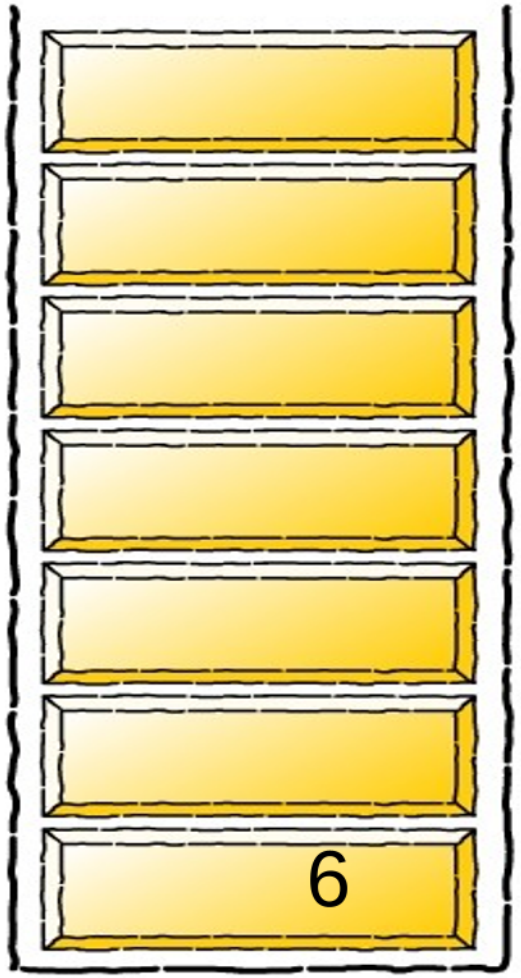


Postfix Expression

3 + *

5 + 40 = €

Evaluation of Postfix Expression

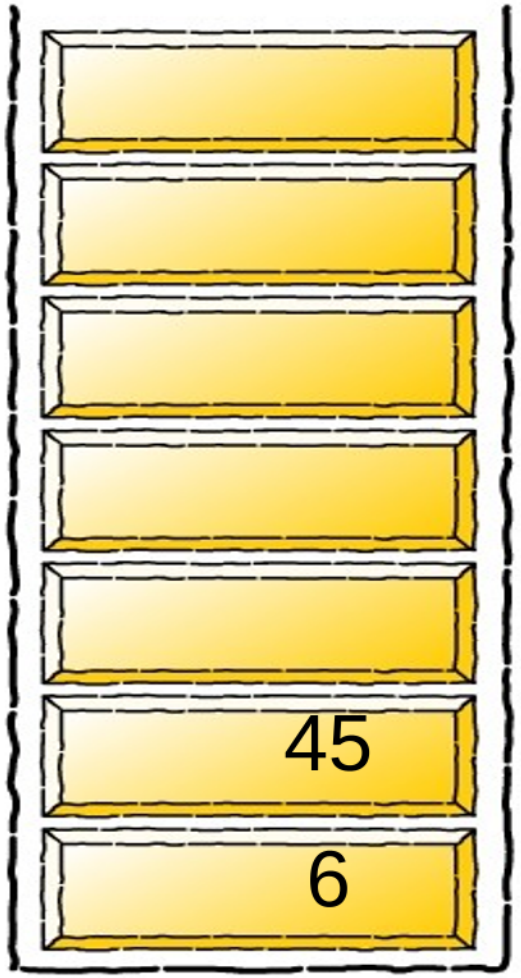


Postfix Expression

3 + *

5 + 40 = 45

Evaluation of Postfix Expression



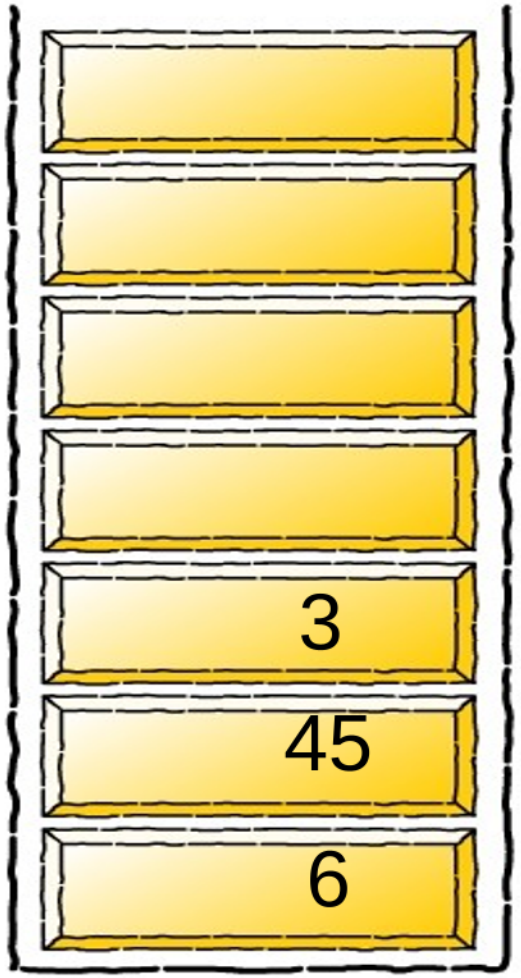
Postfix Expression

3 + *

€ € € = €

Evaluation of Postfix Expression

PES UNIVERSITY

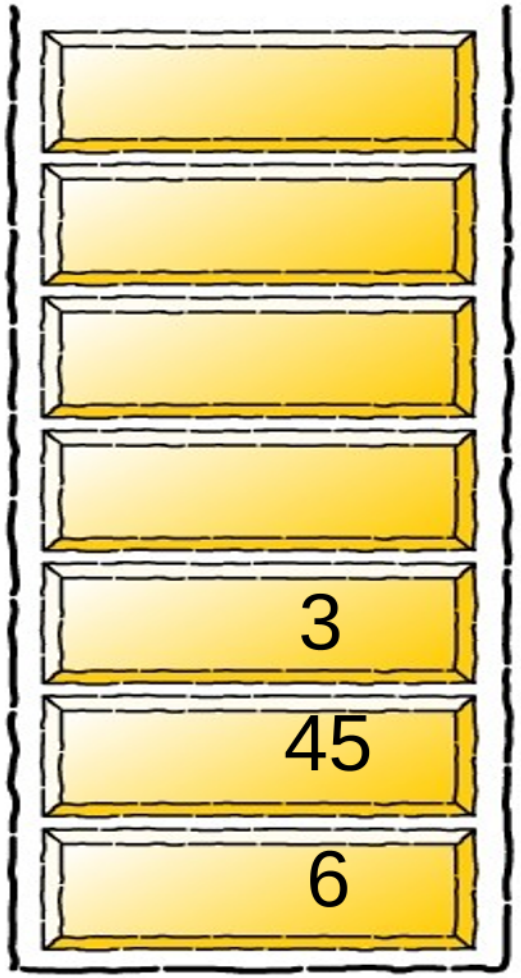


Postfix Expression

+ *

€ € € = €

Evaluation of Postfix Expression

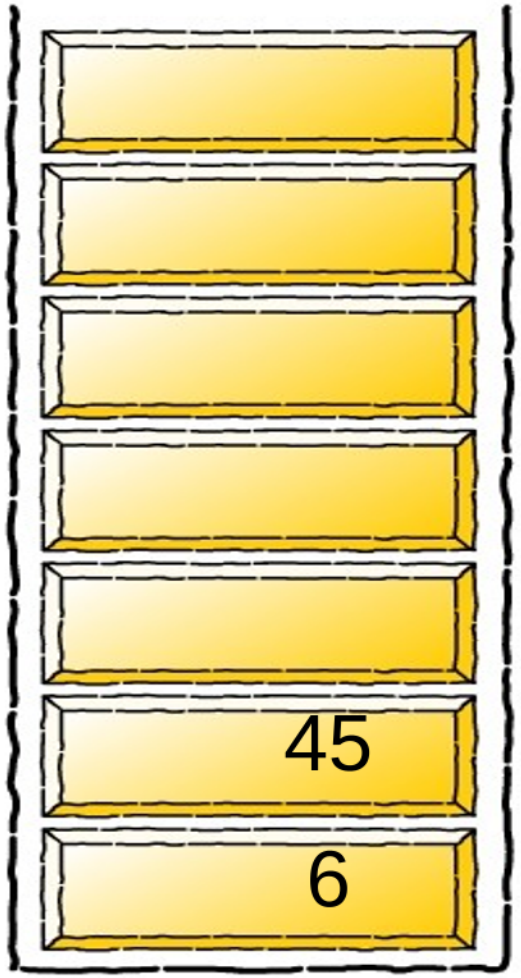


Postfix Expression

*

€ + € = €

Evaluation of Postfix Expression



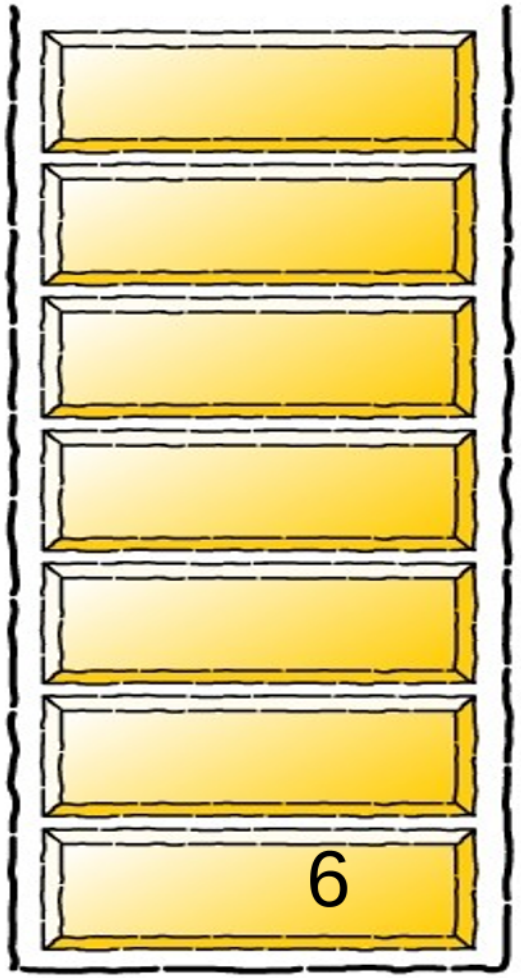
Postfix Expression

*

€ + 3 = €

Evaluation of Postfix Expression

PES UNIVERSITY

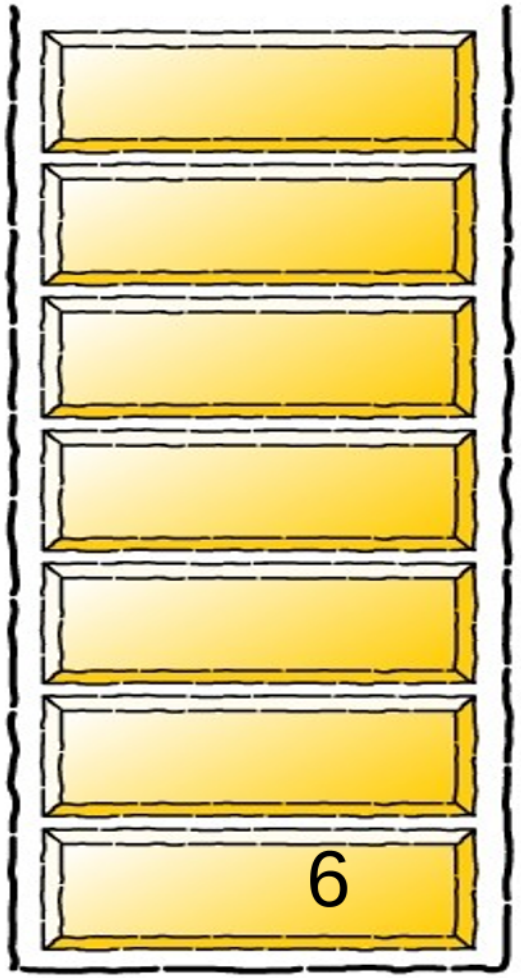


Postfix Expression

*

45 + 3 = €

Evaluation of Postfix Expression

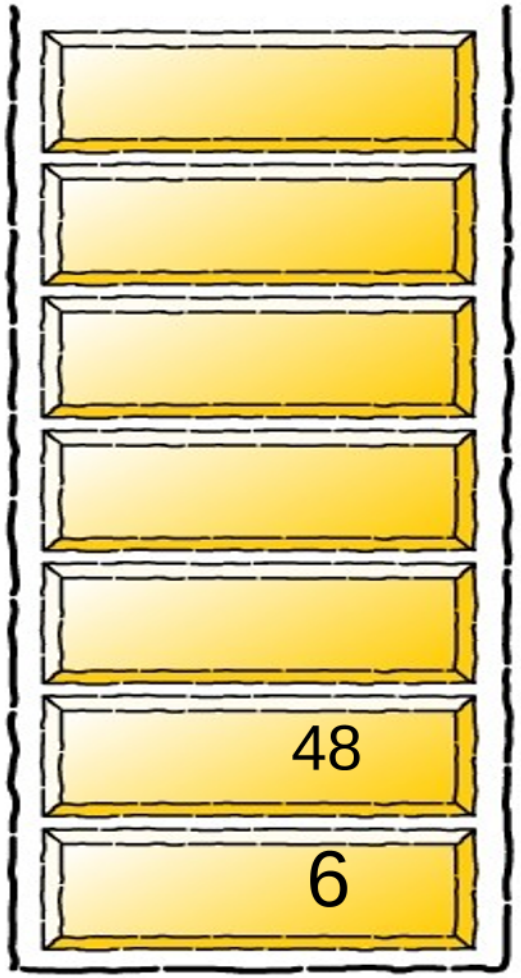


Postfix Expression

*

45 + 3 = 48

Evaluation of Postfix Expression

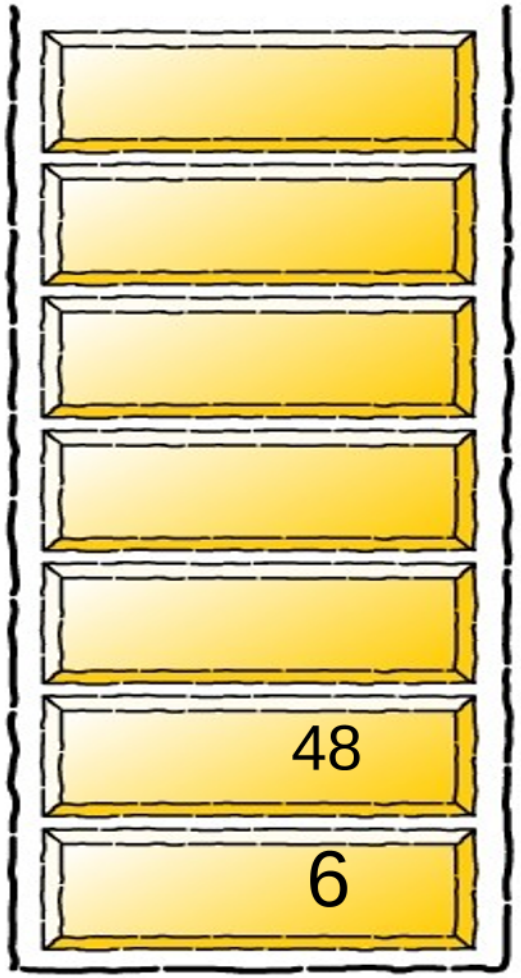


Postfix Expression

*

€ € € = €

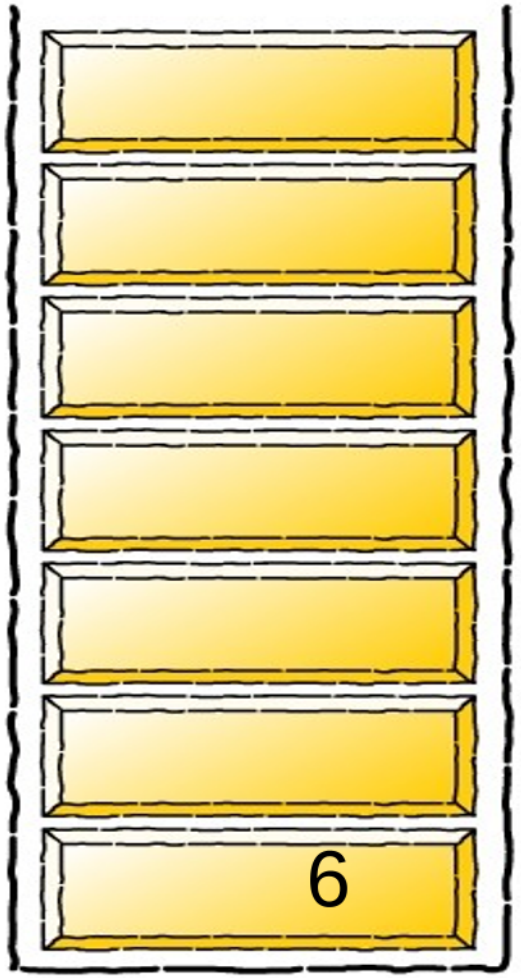
Evaluation of Postfix Expression



Postfix Expression

€ * € = €

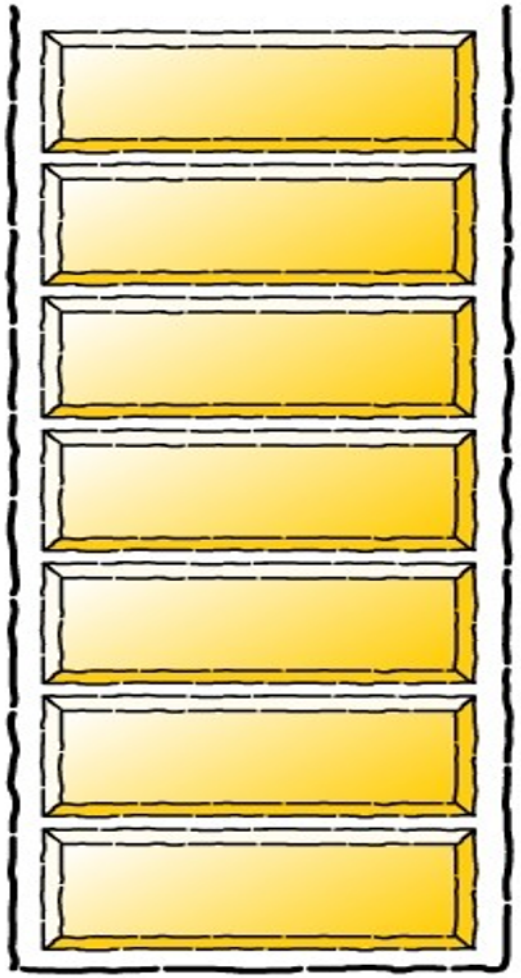
Evaluation of Postfix Expression



Postfix Expression

€ * 48 = €

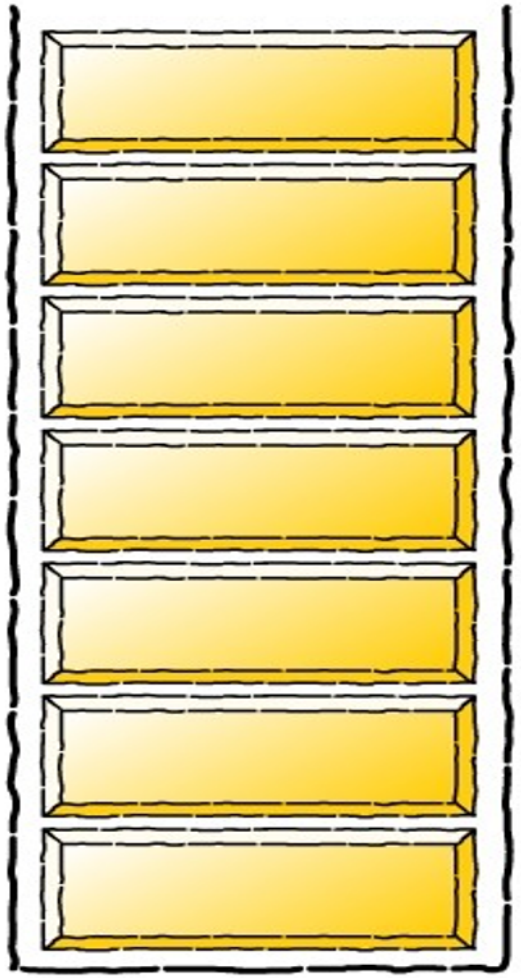
Evaluation of Postfix Expression



Postfix Expression

6 * 48 = €

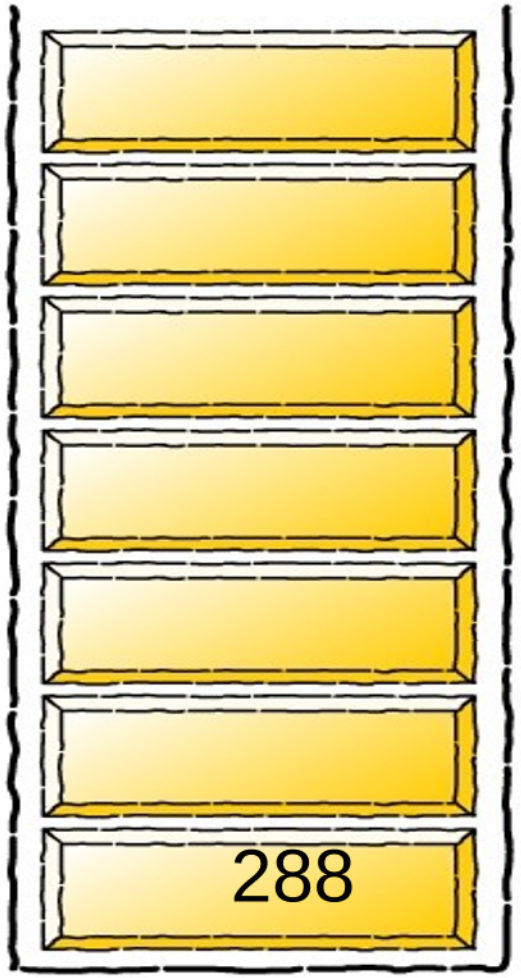
Evaluation of Postfix Expression



Postfix Expression

6 * 48 = 288

Evaluation of Postfix Expression



Postfix Expression

€ € € = €

Evaluation of Postfix Expression

Input	Stack *(top is on the left)*	
2 3 4 * +	*empty*	Push 2
3 4 * +	2	Push 3
4 * +	3 2	Push 4
* +	4 3 2	Pop 4, pop 3, do 3 *4 , push 12
+	12 2	Pop 12, Pop 2, do 2 + 12, push 14
	14	

Input	Stack	
3 4 * 2 5 * +	*empty*	Push 3
4 * 2 5 * +	3	Push 4
* 2 5 * +	4 3	Pop 4, pop 3, do 3*4, Push 12
2 5 * +	12	Push 2
5 * +	2 12	Push 5
* +	5 2 12	Pop 5, Pop 2, do 2*5, Push 10
+	10 12	Pop 10, Pop 12 do 12 + 10, push 22
	22	

Evaluation of Postfix Expression

PES UNIVERSITY

- Problems (A=3,B=2,C=4,D=6)
  - ABC*+D+
  - AB+CD+*
  - AB*CD*+
  - AB+C+D+
  - 20 50 3 6 + * * 300 / 2 -

THANK YOU

Dilip Maripuri
Associate Professor
Department of Computer Applications
dilip.maripuri@pes.edu