Data
Structure

Stack

Instructors:

Md Nazrul Islam Mondal
Department of Computer Science & Engineering
Rajshahi University of Engineering &
Technology Rajshahi-6204

Outline

- Stacks
- Array Representation of Stacks
- Linked Representation of Stacks
- Arithmetic Expression: Polish Notation

Stacks

Stacks

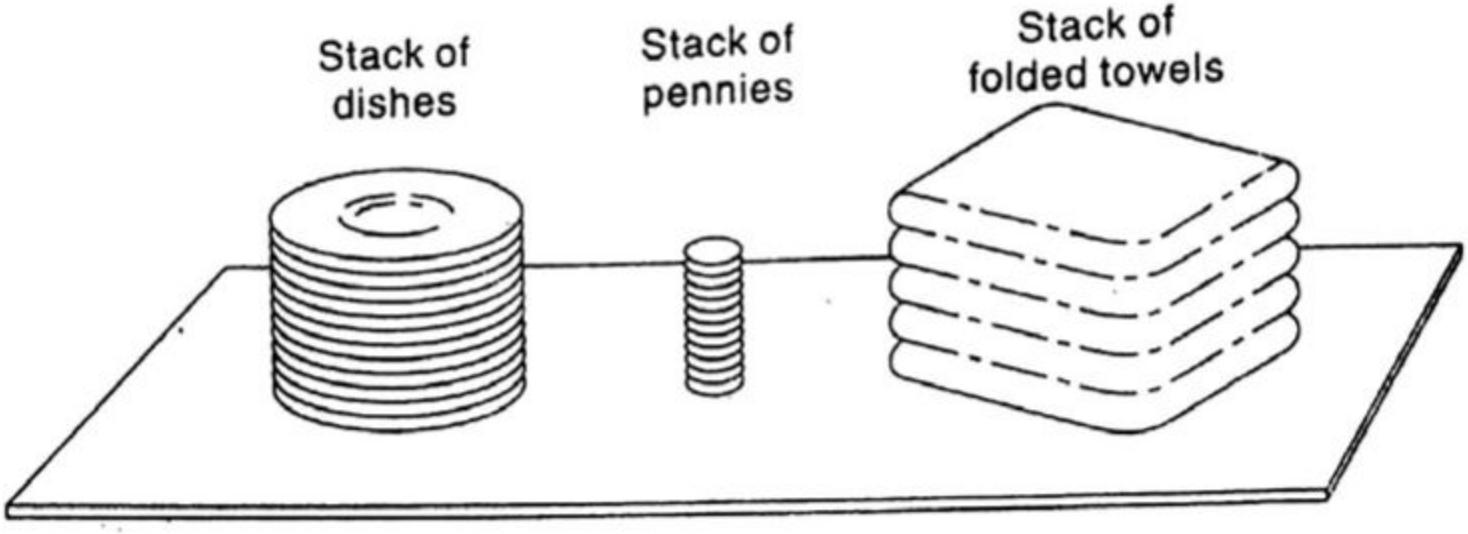


Fig. 6.1

Stacks

- An element is inserted or deleted only at one end, called the **top** of the stack.
- Special terminology is used for two basic operations associate with stack:
  - "Push" is the term used to insert an element into a stack
  - "Pop" is the term used to delete an element from a stack

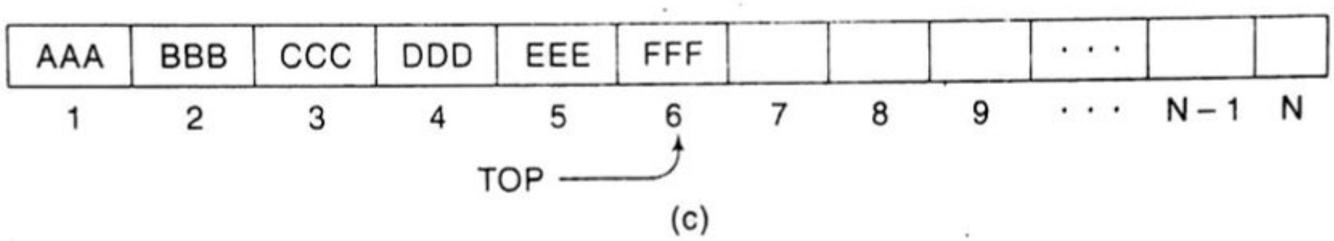


Fig. 6.3 *Diagrams of Stacks*

Stacks

- Postponed Decisions
  - Stacks are frequently used to indicate the order of the processing of data when certain steps of the processing must be postponed until other conditions are fulfilled.
  - Suppose that while processing some project A we are required to move on to project B, whose completion is required in order to complete project A.

Stacks

- Postponed Decisions

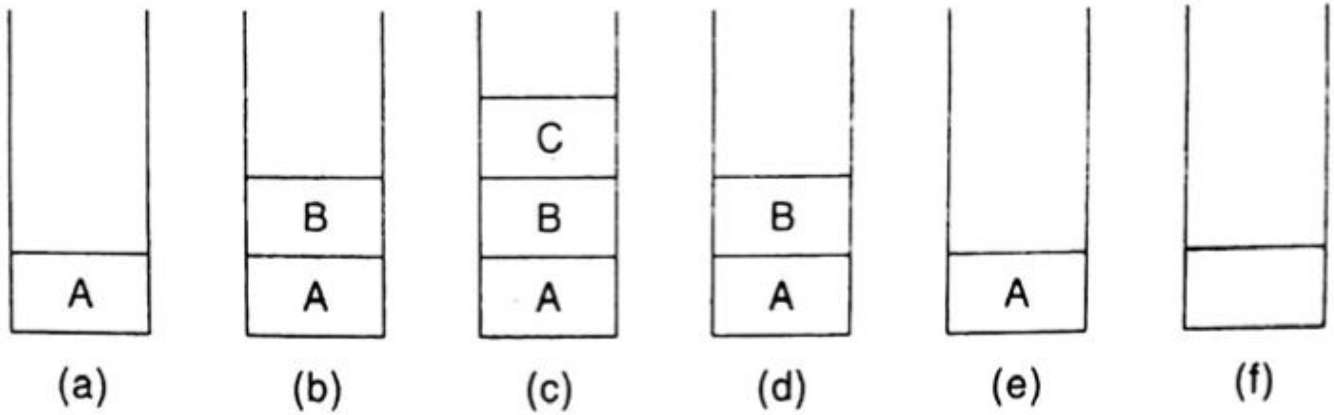


Fig. 6.4

Array Representation of Stack

Array Representation of Stack

- A linear array STACK
- A pointer variable TOP, which contains the location of the top element of the stack
- A variable MAXSTK which gives the maximum number of elements that can be held by the stack.
- The condition TOP=0 or TOP=NULL will indicate that the stack is empty.

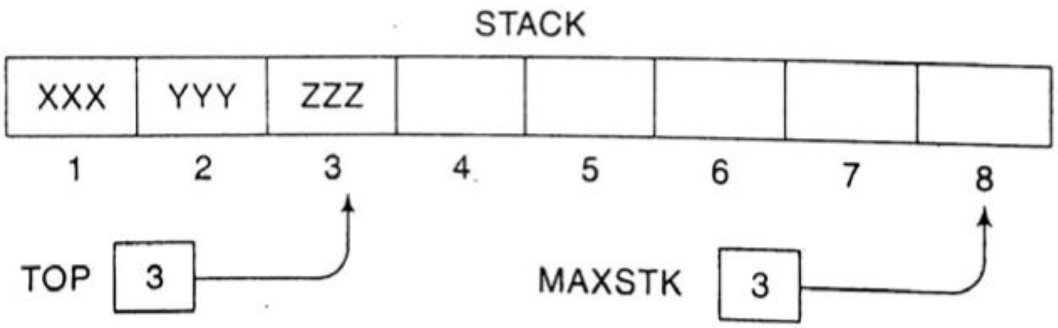
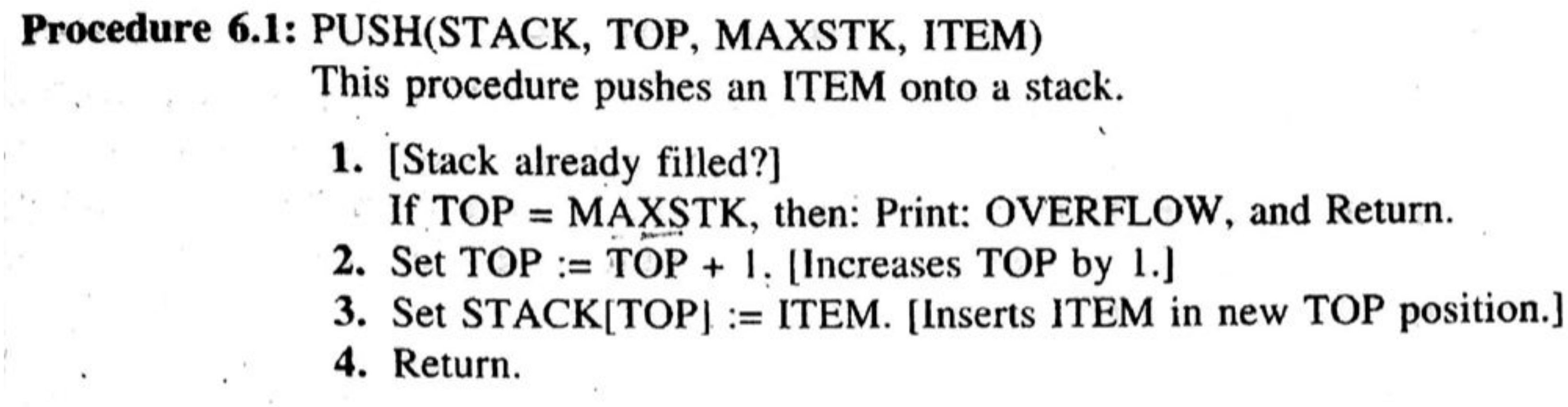


Fig. 6.5

Array Representation of Stack

- PUSH – the operation of adding(pushing) an item into a stack
- POP – the operation of removing(Popping) an item from a stack
- PUSH is associate with OVERFLOW.
- POP is associate with UNDERFLOW.

Array Representation of Stack

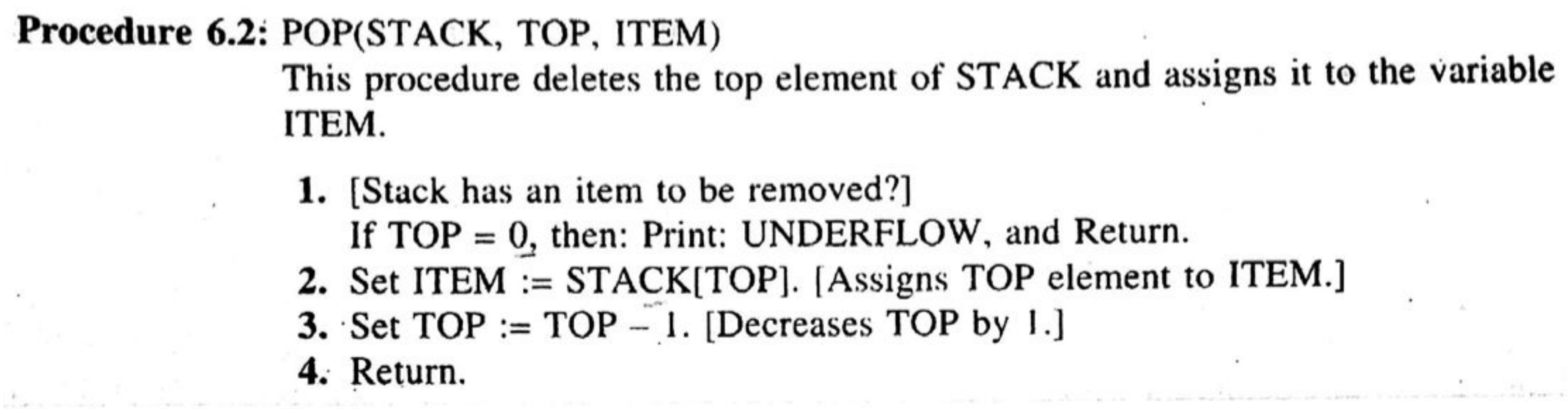


Procedure 6.1: PUSH(STACK, TOP, MAXSTK, ITEM)

This procedure pushes an ITEM onto a stack.

```
1. [Stack already filled?]
   If TOP = MAXSTK, then: Print: OVERFLOW, and Return.
2. Set TOP := TOP + 1. [Increases TOP by 1.]
3. Set STACK[TOP] := ITEM. [Inserts ITEM in new TOP position.]
4. Return.
```

Array Representation of Stack



Procedure 6.2: POP(STACK, TOP, ITEM)

This procedure deletes the top element of STACK and assigns it to the variable ITEM.

```
1. [Stack has an item to be removed?]
   If TOP = 0, then: Print: UNDERFLOW, and Return.
2. Set ITEM := STACK[TOP]. [Assigns TOP element to ITEM.]
3. Set TOP := TOP – 1. [Decreases TOP by 1.]
4. Return.
```

Array Representation of Stack

- Minimizing Overflow
  - There is **no direct control** by the programmer.
  - But programmer sets the amount of memory space reserved for each stack, and this choice does influence the number of time overflow may occur.
  - **Time-space Tradeoff**
    - Reserving a great deal of space for each stack will decrees the number of time overflow may occur; however, this may be an expensive use of the space if most of the space is seldom used.
    - On the other hand, reserving a small amount of space for each stack may increase the number of time overflow occurs; and the time required for resolving an overflow, such as by adding space to the stack may be more expensive than the space saved.
  - Various techniques have been developed which modify the array representation of stacks so that the amount of space reserved for more than one stack may be more efficiently used.

Linked Representation of Stacks

Linked Representation of Stacks

- Used One-way List or Singly Linked List
- The START pointer of the linked list behaves as the TOP pointer variable of the stack.

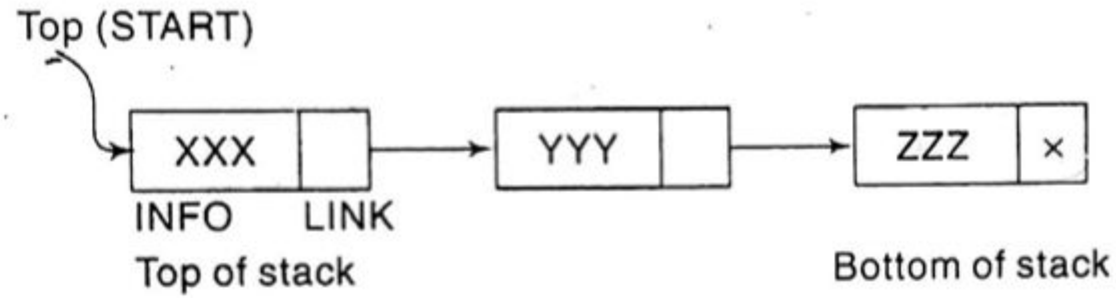


Fig. 6.7

Linked Representation of Stacks

- POP and PUSH operation

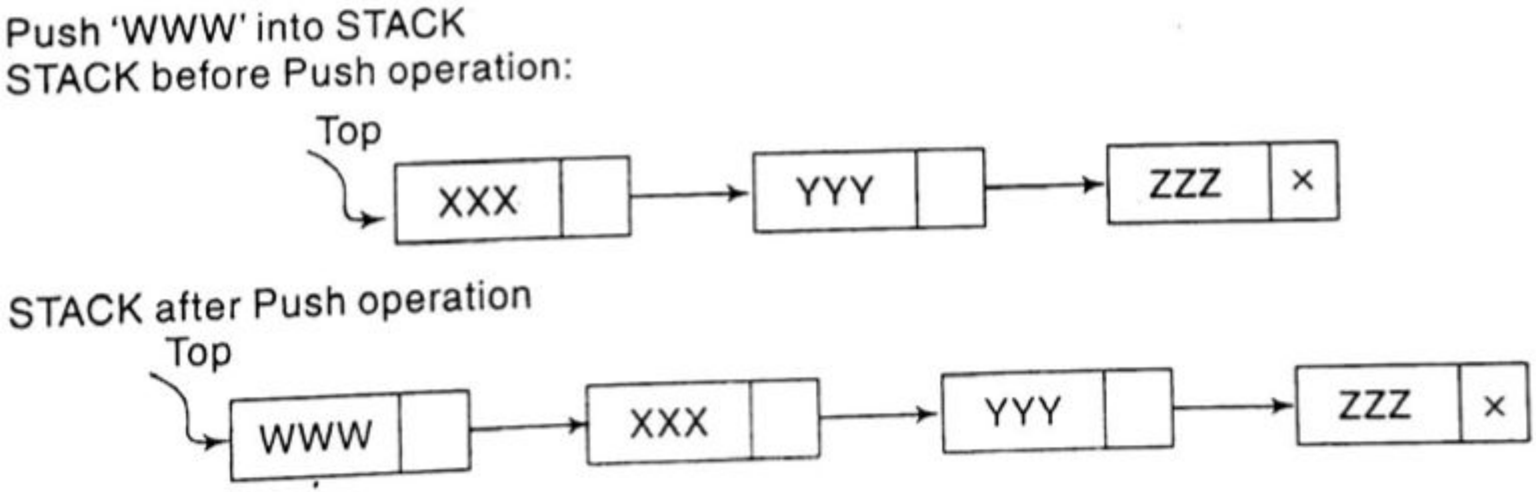


Fig. 6.8

Linked Representation of Stacks

- POP and PUSH operation

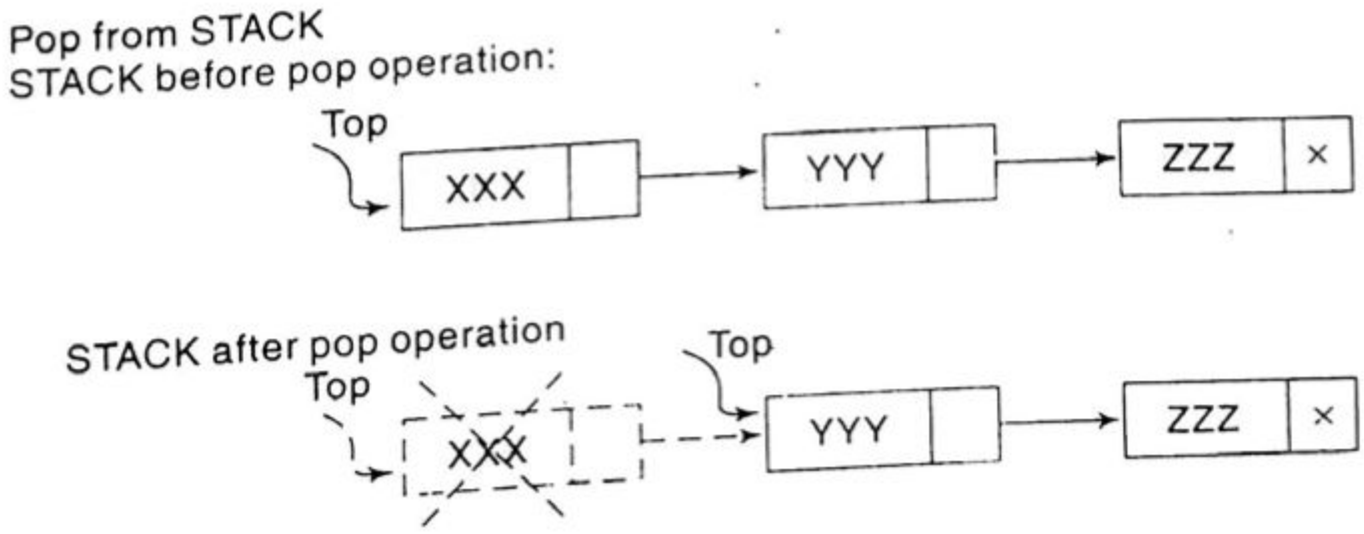
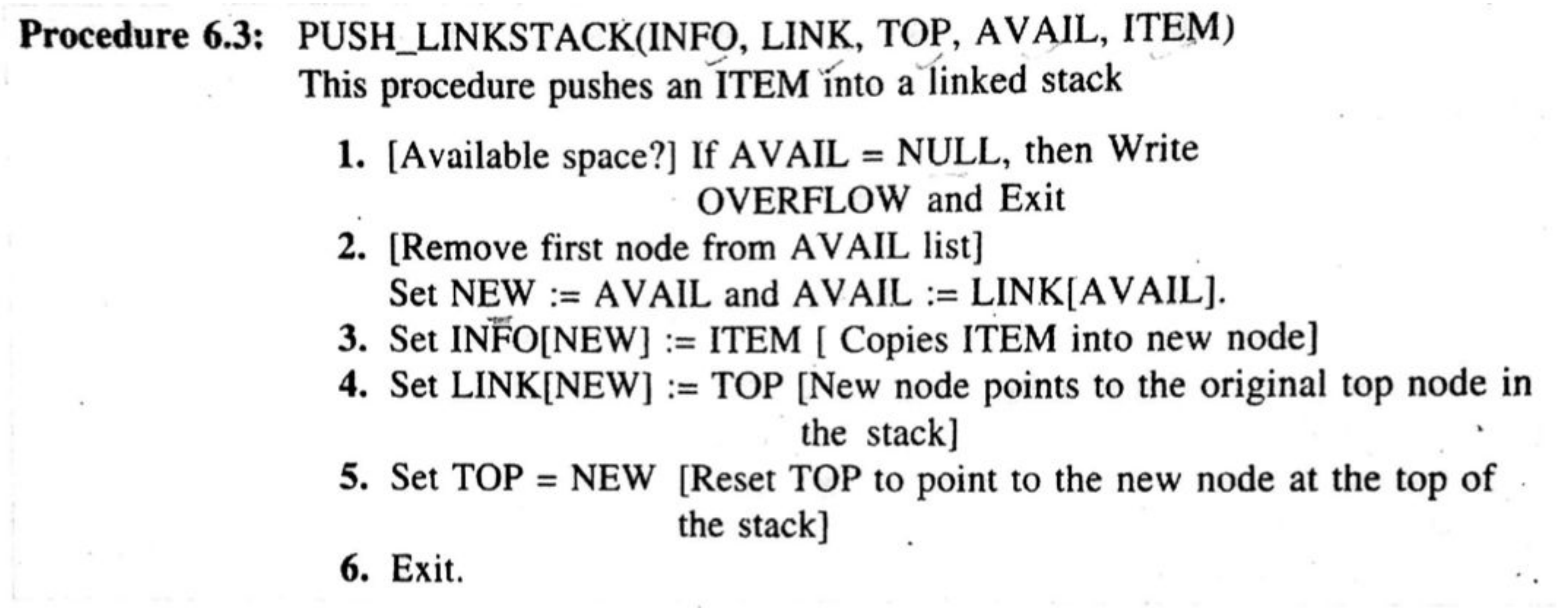


Fig. 6.9

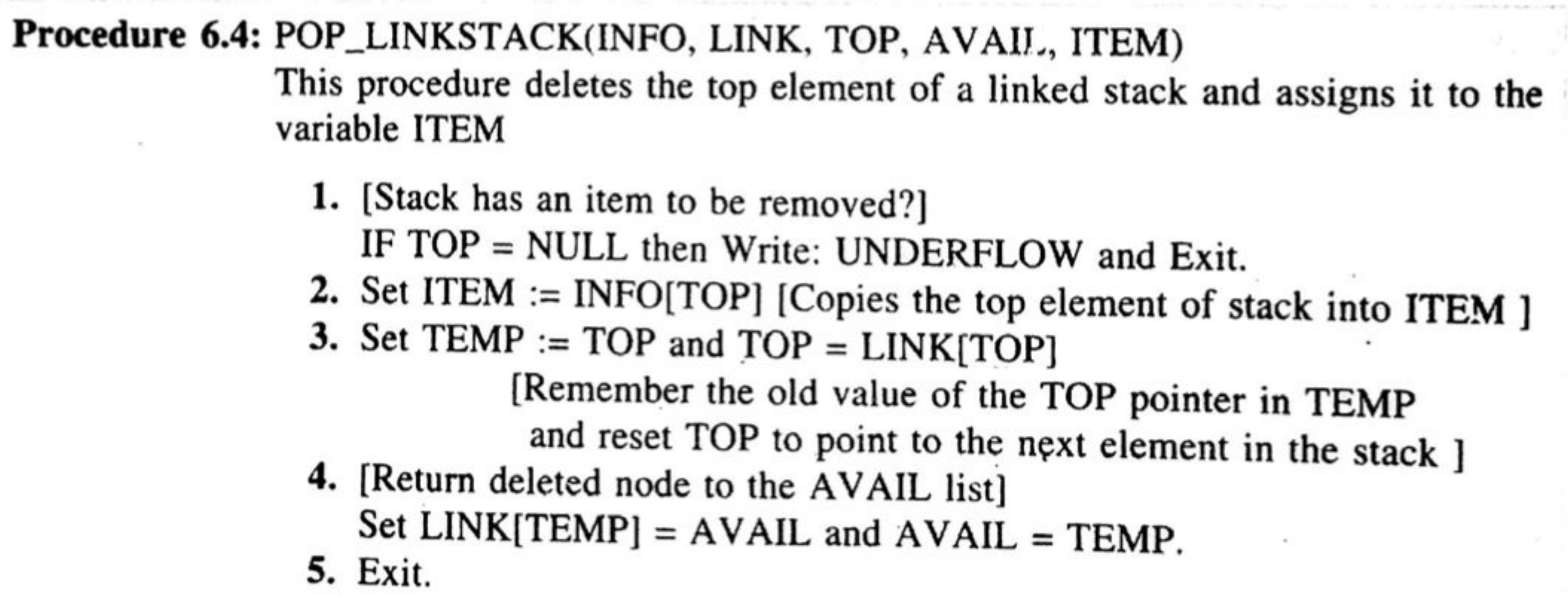
Linked Representation of Stacks



Procedure 6.3: PUSH_LINKSTACK(INFO, LINK, TOP, AVAIL, ITEM)
This procedure pushes an ITEM into a linked stack

```
1. [Available space?] If AVAIL = NULL, then Write
                                OVERFLOW and Exit
2. [Remove first node from AVAIL list]
   Set NEW := AVAIL and AVAIL := LINK[AVAIL].
3. Set INFO[NEW] := ITEM [ Copies ITEM into new node]
4. Set LINK[NEW] := TOP [New node points to the original top node in
                                the stack]
5. Set TOP = NEW [Reset TOP to point to the new node at the top of
                  the stack]
6. Exit.
```

Linked Representation of Stacks



Procedure 6.4: POP_LINKSTACK(INFO, LINK, TOP, AVAIL, ITEM)

This procedure deletes the top element of a linked stack and assigns it to the variable ITEM

```
1. [Stack has an item to be removed?]
   IF TOP = NULL then Write: UNDERFLOW and Exit.
2. Set ITEM := INFO[TOP] [Copies the top element of stack into ITEM ]
3. Set TEMP := TOP and TOP = LINK[TOP]
              [Remember the old value of the TOP pointer in TEMP
               and reset TOP to point to the next element in the stack ]
4. [Return deleted node to the AVAIL list]
   Set LINK[TEMP] = AVAIL and AVAIL = TEMP.
5. Exit.
```

Arithmetic Expression: Polish Notation

Arithmetic Expression: Polish Notation

- An application of Stack
- Let Q be an arithmetic expression involving constants and operations.
- The binary operations in Q may have different levels of precedence.

Highest	:	Exponentiation(↑)
Next highest	:	Multiplication(*) and division(/)
Lowest	:	Addition(+) and substraction(-)

Arithmetic Expression: Polish Notation

- Infix Notation:

  A+B   A-B   A/B   A*C

- Problem:
  - We must distinguish between (A+B)*C and A+(B*C) by using either parentheses or some operator-precedence convention.

- To solve this problem:
  - **Polish Notation**, named after the Polish Mathematician Jan Lukasiewicz
  - **Polish Notation:** The operator symbol is placed before its two operands.

    +AB -AB *AB /AB
  - This notation is also called **Prefix Notation.**

Arithmetic Expression: Polish Notation

- Translate the following infix expression into Polish Notation:

(A+B)*C = [+AB]*C =*+ABC

A+(B*C) = A+[*BC] = +A*BC

(A+B)/(C-D) = [+AB]/[-CD] = /+AB-CD

Arithmetic Expression: Polish Notation

- **Reverse Polish Notation:** the operator symbol is placed after its two operands.

  AB+ CD- EF* GH/

- This notation is frequently called **postfix (or suffix) notation.**

Arithmetic Expression: Polish Notation

- Evaluate an arithmetic expression written in infix notaion in two steps
  - Converts the expression to postfix notation (using STACK)
  - Evaluate the postfix expression (using STACK)

Arithmetic Expression: Polish Notation

- Transforming Infix Expressions into Postfix Expressions
  - Consider the following arithmetic infix expression Q:

    Q: A+(B*C-(D/E↑F)*G)*H

Initial Step: Push left parenthesis onto STACK

Symbol Scanned		STACK	Expression P
		(	

Add a right parenthesis to the end of Q

Q: A+(B*C-(D/E↑F)*G)*H)

Arithmetic Expression: Polish Notation

Symbol Scanned		STACK	Expression P
		(	
(1)	A	(	A
(2)	+	(+	A
(3)	(	(+ (	A
(4)	B	(+ (	A B
(5)	*	(+ (*	A B
(6)	C	(+ (*	A B C

Q: A+(B*C-(D/E↑F)*G)*H)

Operands Expression P
Operators and bracket STACK

Arithmetic Expression: Polish Notation

Symbol Scanned		STACK	Expression P
(6)	C	(+ (*	A B C
(7)	-	(+ (-	A B C *

Q: A+(B*C-(D/E↑F)*G)*H)

7. The subtraction operator send **multiplication operator** from STACK to Expression P befor it is pushed onto STACK
   - (*) operator precedence is higher than (-) operator

Arithmetic Expression: Polish Notation

Symbol Scanned		STACK	Expression P
(7)	-	(+ (-	A B C *
(8)	(	(+ (- (	A B C *
(9)	D	(+ (- (	A B C * D
(10)	/	(+ (- (/	A B C * D
(11)	E	(+ (- (/	A B C * D E
(12)	↑	(+ (- (/ ↑	A B C * D E

Q: A+(B*C-(D/E↑F)*G)*H)

12. Division and Multiplication operator precedence is equal

Arithmetic Expression: Polish Notation

Symbol Scanned		STACK	Expression P
(12)	↑	(+ (- (/ ↑	A B C * D E
(13)	F	(+ (- (/ ↑	A B C * D E F
(14)	)	(+ (-	A B C * D E F ↑ /
(15)	*	(+ (- *	A B C * D E F ↑ /
(16)	G	(+ (- *	A B C * D E F ↑ / G

Q: A+(B*C-(D/E↑F)*G)*H)

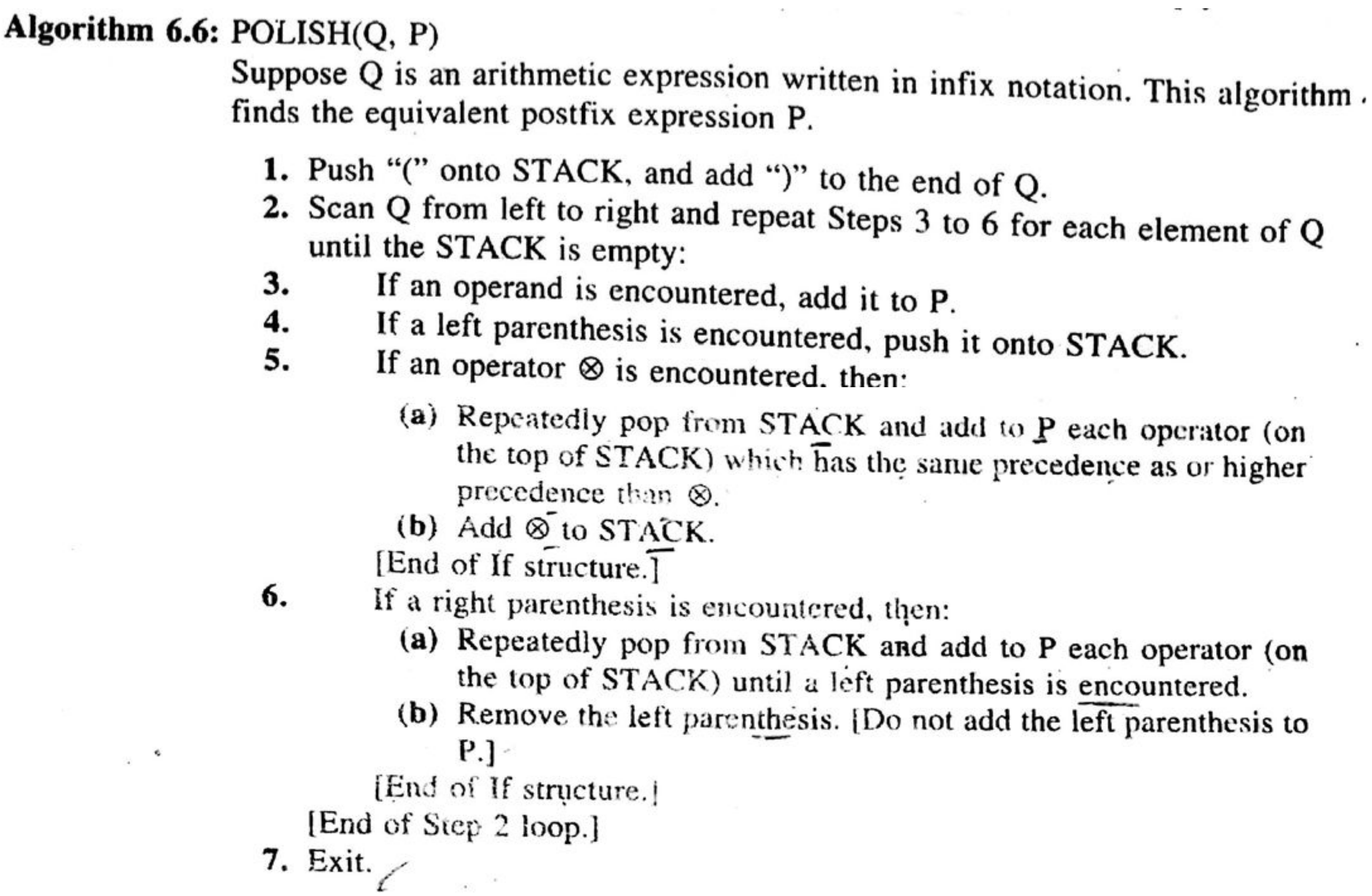
14. The right parenthesis sends ↑ and / from STACK to P, and then removes the left parenthesis from the top of STACK.
15. - precedence is less than *

Arithmetic Expression: Polish Notation

Symbol Scanned		STACK	Expression P
(16)	G	(+ (- *	A B C * D E F ↑ / G
(17)	)	(+	A B C * D E F ↑ / G * -
(18)	*	(+ *	A B C * D E F ↑ / G * -
(19)	H	(+ *	A B C * D E F ↑ / G * - H
(20)	)		A B C * D E F ↑ / G * - H * +

Q: A+(B*C-(D/E↑F)*G)*H)

Arithmetic Expression: Polish Notation



```
Algorithm 6.6: POLISH(Q, P)
    Suppose Q is an arithmetic expression written in infix notation. This algorithm
    finds the equivalent postfix expression P.

    1. Push "(" onto STACK, and add ")" to the end of Q.
    2. Scan Q from left to right and repeat Steps 3 to 6 for each element of Q
       until the STACK is empty:
    3.      If an operand is encountered, add it to P.
    4.      If a left parenthesis is encountered, push it onto STACK.
    5.      If an operator $\otimes$ is encountered, then:
              (a) Repeatedly pop from STACK and add to P each operator (on
                  the top of STACK) which has the same precedence as or higher
                  precedence than $\otimes$.
              (b) Add $\otimes$ to STACK.
            [End of If structure.]
    6.      If a right parenthesis is encountered, then:
              (a) Repeatedly pop from STACK and add to P each operator (on
                  the top of STACK) until a left parenthesis is encountered.
              (b) Remove the left parenthesis. [Do not add the left parenthesis to
                  P.]
            [End of If structure.]
       [End of Step 2 loop.]
    7. Exit.
```

Arithmetic Expression: Polish Notation

- Evaluation of a Postfix Expression
  - Consider the following arithmetic expression P written in postfix notation

    P: 5, 6, 2, +, *, 12, 4, /, -

  - Add a right parenthesis at the end of P

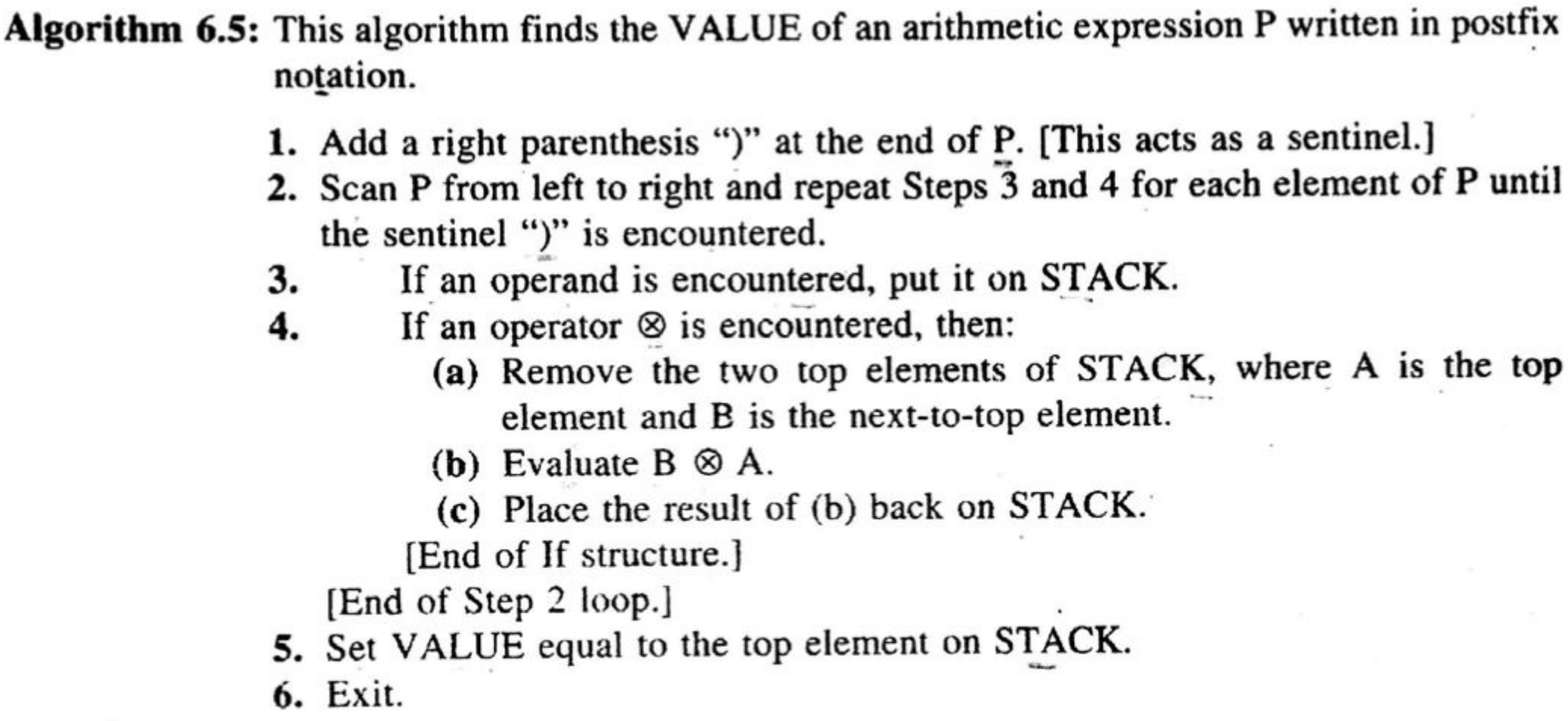
    P: 5, 6, 2, +, *, 12, 4, /, - ,)

Arithmetic Expression: Polish Notation

Symbol Scanned		STACK
(1)	5	5
(2)	6	5, 6
(3)	2	5, 6, 2
(4)	+	5, 8
(5)	*	40
(6)	12	40, 12
(7)	4	40, 12, 4
(8)	/	40, 3
(9)	-	37
(10)	)	

P: 5, 6, 2, +, *, 12, 4, /, - ,)

Arithmetic Expression: Polish Notation



Algorithm 6.5: This algorithm finds the VALUE of an arithmetic expression P written in postfix notation.

```
1. Add a right parenthesis ")" at the end of P. [This acts as a sentinel.]
2. Scan P from left to right and repeat Steps 3 and 4 for each element of P until
   the sentinel ")" is encountered.
3.       If an operand is encountered, put it on STACK.
4.       If an operator ⊗ is encountered, then:
           (a) Remove the two top elements of STACK, where A is the top
               element and B is the next-to-top element.
           (b) Evaluate B ⊗ A.
           (c) Place the result of (b) back on STACK.
         [End of If structure.]
   [End of Step 2 loop.]
5. Set VALUE equal to the top element on STACK.
6. Exit.
```

END