Date 1/01/24
Page

Stack operation

To use a stack efficiently, we need check status of stack as well. For the same purpose, the following functionality is added to stacks:

* top() - get the top data element of stack without removing it
* isfull() - check if stack is full
* isempty() - checks if stack is empty.

→ push operation implies the insertion of a new element into stack
→ A new element is always inserted from the topmost position of the stack.

Top = max - 1
if this condition goes false
* it means the stack is full
* And no more elements can be inserted
* And even if we try to insert the element, a stack overflow message will be displayed.

```
begin
  if Stack is full        --> Top == max - 1
    return
  end if
  else
    increment to          --> Top = Top + 1
    Stack[top] assign value
    end else
end procedure.
```

push

Date / /
Page

→ Pop meant to delete an element from the stack.

→ Before deleting an element, make sure to check if the stack top is null.
ex TOP = NULL

If this condition goes true.
. . it means the stack is empty.
And no deletion operation can be performed,
→ and even if we try to delete then the stack underflow message will b generated.

```
begin
  if stack is empty --> TOP == NULL
    return
  end if
  else
    store value of stack[top]
    decrement top
    return value
    end else
    end procedure

TOP()
    begin
      return stack[top]
      end procedure
```

isempty()
returns true if the stack is empty
else false.

```
begin
    if top<1
        return true
    else
        return false
end procedure
```

```
display()
begin
  if top==-1
  Print Stack is empty

for(i=top; i>0; i--)
  Print stack element.
```

infix to postfix expression

Algorithm used
postfix
step1: add ")" to the end of the infix expression
step2: push(onto the stack.
step3: repeat until each character in the infix notation scanned
- if an operand is encountered add it postfix expression.

~~if an operand~~
if a "(" is encountered, push it on the stack
if a ")" is encountered, then
a. Repeatedly pop from stack and add it to the postfix expression until a "(" is encountered
b. Discard th "(" that is. remove the (from stack and do not add it to the postfix expression.
if an operator o is encountered then

a. Repeatedly pop from stack and add each operator to the post fix expression which the same precedence or a higher precedence then o
b. Push the operator to stack
[end of if]
step4: Repeatedly pop from the stack and add it to postfix expression until the stack is empty.
step5: exit

prefix

step1: Reverse the infix string. note that while reversing the string you must interchange left & right parentheses
step2: obtain the postfix expression of the infix expression step1.
step3: Reverse the postfix expression to get the prefix expression.