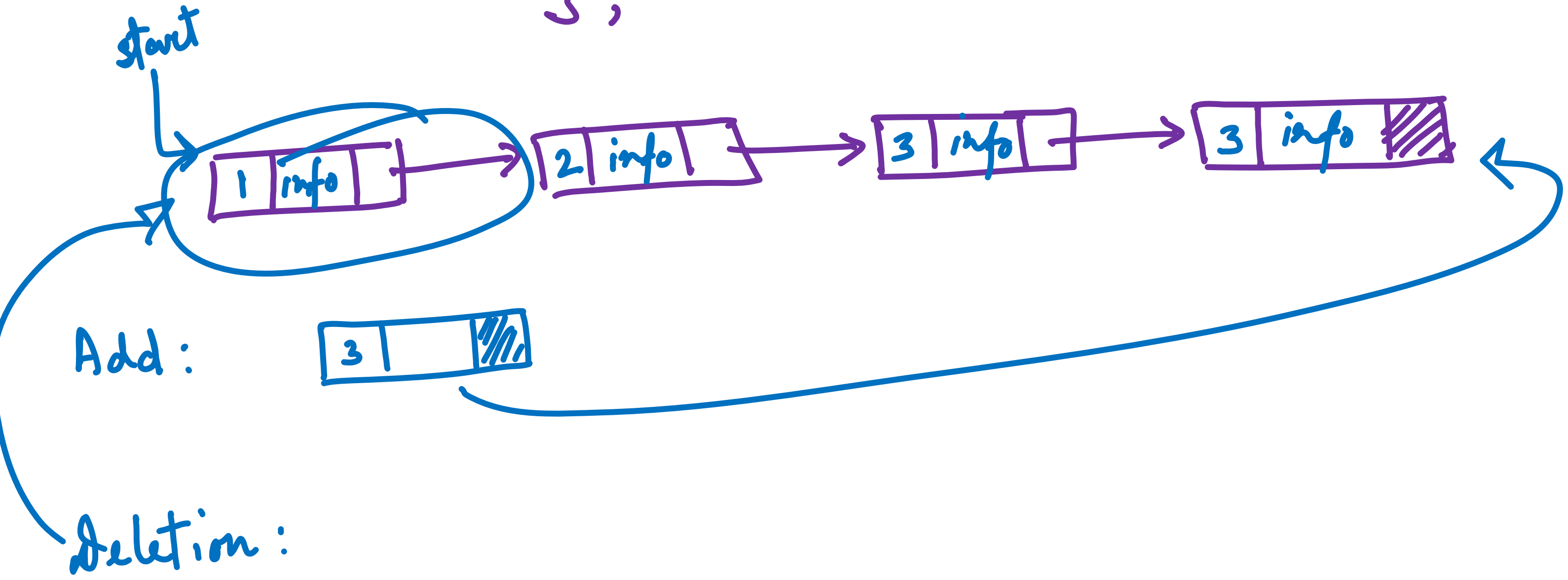
Priority Queue

Dated 02/12/2021

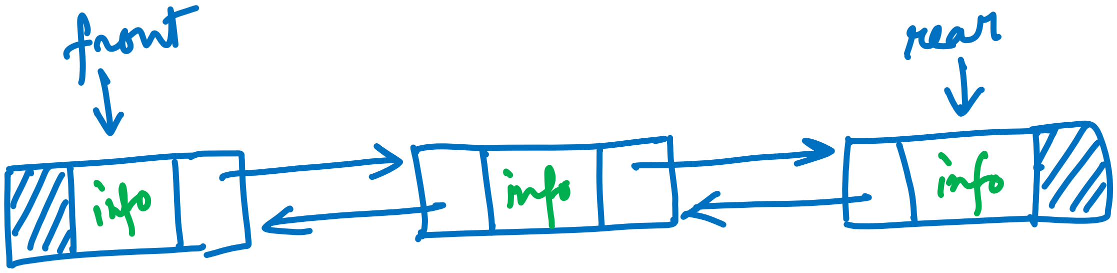
```
struct pq
    {   int priority;
        int data;
        struct pq * next;
    };
```



DEQUE (Double Ended Queue)

(1) Input restricted → elements can be added only at the rear end but, can be deleted from both sides

(2) Output restricted → elements can be added on both sides but, can be deleted only at the start/front of the queue



Insertion at the front end :-

```
If newnode == NULL then printf("Overflow"),
else   if front == NULL then rear = front = newnode;
       else
         {   newnode -> next = front ;
             front -> prev = newnode ;
             newnode -> prev = NULL ;
             front = newnode ;
         }
```

Insertion at rear end :-

```
If newnode == NULL then printf("Overflow");
else   if rear == NULL then front = rear = newnode ;
       else
         {   newnode -> prev = rear ,
             rear -> next = newnode ;
             newnode -> next = NULL ;
             rear = newnode ;
         }
```

Deletion from the front end :—

```
If front == NULL then printf("Underflow");
else
        {       temp = front ;
                front = front → next ,
                If front == NULL   then  rear = NULL;
                else
                     front → prev = NULL ;
                free (temp) ;
        }
```

Deletion from the rear end :—

```
If front == NULL  then  printf("Underflow");
else
        {       Temp = rear ;
                rear = rear → prev ;
                If rear == NULL  then  front = NULL;
                else
                     rear → next = NULL ;
                free (temp) ;
        }
```

Polish Notation Conversion & Evaluation

Dated on. 04|12|2021 (Morning)

Polish mathematician "Jan Lukasiewich" came with a new technique for representing arithmetic expressions, where operators will be placed either before or after the operands. These denotions is termed as Polish notation or Reverse Polish notation in his honor.

Infix : A + B

Prefix : + AB $\Longrightarrow$ PN : Polish Notation

Postfix : AB + $\Longrightarrow$ RPN : Reverse Polish Notation

Ex-1 Infix : $(A+B)/C * D - E$ where A, B, C, D, E are arbitrary constants.

Prefix : $(+AB)/C * D - E$

$= (/+ABC) * D - E$

$= (*/+ABCD) - E$

$= -*/+ABCDE \Rightarrow$ Polish Notation "PN"

Priority Levels

Level 2 :	^ (exponentiation)	
Level 1 :	* (Multiplication)	/ (Division)
Level 0 :	+ (Addition)	- (Subtraction)

priority ⇑

Postfix : $(AB+)/C * D - E$

$= (AB+C/) * D - E$

$= (AB+C/D*) - E$

$= AB+C/D*E-$

$\Longrightarrow$ Reverse Polish Notation
"RPN"

Example-2 :

Infix : $A+B/C-D*E+F$

Prefix : $A+(/BC)-(*DE)+F$

$= (+A/BC)-(*DE)+F$

$= (-+A/BC*DE)+F$

$= +-+A/BC*DEF$

$\Downarrow$

"PN"

Postfix. $A + (BC/) - (DE*) + F$

$= (ABC/+) - (DE*) + F$

$= (ABC/+DE*-) + F$

$= ABC/+DE*-F+$

$\Downarrow$

"RPN"

Conversion of infix expression into a postfix form

Step-1 : Add the unique symbol into the stack and at the end of the array infix.

Step-2 : Scan the symbol of array infix one by one from left to right.

Step-3 : If the symbol is left parenthesis '(' then add it to the stack.

Step-4 : If the symbol is operand then add it to array postfix.

Step-5 : (i) If the symbol is operator, then pop the operators which have the same precedence or higher precedence than the operator which occurred.

(ii) Add the popped operators one by one to array postfix.

(iii) Add the scanned symbol operator into the stack.

Step-6 : (i) If the symbol is right parenthesis ')' then pop all operators from the stack until '(' is popped up from the stack. Add the popped ones into the array postfix.

(ii) Remove the '(' from the stack.

Step-7 : If the symbol is '#' then POP all symbols from the stack and add them to array postfix except '#'.

Step-8 : Do the same process until '#' comes in scanning of array infix.

Let us take an infix expression and convert it into postfix form following the above mentioned steps -

Dated on 04/12/2021 (Evening)

A * (B + C ^ D) - E ^ F * (G / H)

$\boxed{A * (B + C ^ D) - E ^ F * (G / H) \#}$

Array Infix	Operator Stack	Postfix array
A	#	A
*	# *	A
(	# * (	A
B	# * (	A B
+	# * (+	A B
C	# * (+	A B C
^	# * (+ ^	A B C
D	# * (+ ^	A B C D
)	# *	A B C D ^ +
-	# -	A B C D ^ + *
E	# -	A B C D ^ + * E
^	# - ^	A B C D ^ + * E
F	# - ^	A B C D ^ + * E F
*	# - *	A B C D ^ + * E F ^
(	# - * (	A B C D ^ + * E F ^

G	# - * (	ABCD^+*EF^G
/	# - * (/	ABCD^+*EF^G
H	# - * (/	ABCD^+*EF^GH
)	# - *	ABCD^+*EF^GH/
#	#	ABCD^+*EF^GH/*-

The conversion result from "infix To postfix" form is $\boxed{\text{ABCD\^{}+*EF\^{}GH/*-}}$