Stacks & Queues, July 24

24 July 2020 19:11

1. Infix, Prefix, Postfix Expressions //Done
2. Postfix Evaluate //Done
3. Infix to Postfix Conversion //Done
4. Queues //Done
5. Circular Queues //Monday

Tasks

1. Linked Lists 101
2. Infix to Prefix
3. Prefix to Postfix
4. Prefix Evaluation
5. Infix to Postfix with brackets

6. LinkedList implementation of Queue

Infix Expression $\rightarrow$ $a + b$

$+ \rightarrow$ operator

$a, b \rightarrow$ operands

$$\boxed{\text{<operand> <operator> <operand>}}$$

Precedance

BODMAS

(left → right)

$a \div b \rightarrow a \times b$

$\overbrace{3 + 4 * 5} \rightarrow 23$

$\underbrace{3 + 4} * 5 \rightarrow 35$

$2 \wedge \underbrace{3 \wedge 2} \rightarrow 2 \wedge 9 \rightarrow 512$ [Right to left]

~~$2 \wedge 3 \wedge 2 \rightarrow 8 \wedge 2 \Rightarrow 64$~~

Prefix Expression (+ a b)

<operator> <operand> <operand>

3 + (4 * 5)

$\downarrow$

$\underbrace{3}_{a} \underbrace{+}_{(op)} \underbrace{(* 4 5)}_{b}$

$\boxed{+ \; 3 * 4 5}$ → Prefix expression of

Postfix Expression → ab +

$\underset{\text{Left}}{<operand>} \; \underset{\text{Right}}{<operand>} \; <operator>$

$3 + \underbrace{4 * 5}$

/*package whatever //do not write package name here
*/

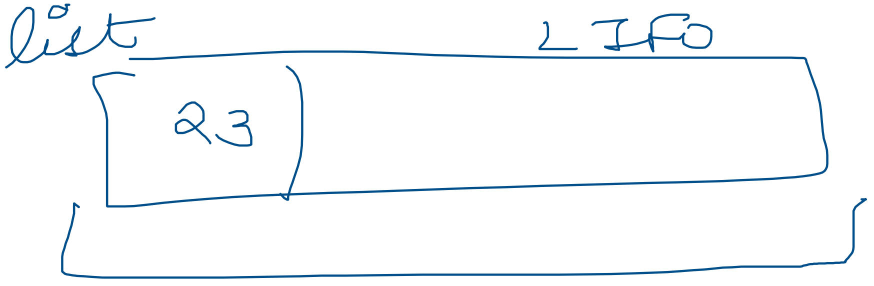
$\boxed{345*+} \rightarrow$ Postfix

Postfix Evaluation

345* +

ab <operator>

list LIFO



4*5

Stack?

Infix to Postfix

1st

$3 + \overbrace{4*5}$

2nd

3*4 +5

34* +5

```
import java.util.*;
import java.lang.*;
import java.io.*;

class GFG {
        public static void main (String[] args) {
                //code
                Scanner sc = new Scanner(System.in);
                int t = Integer.parseInt(sc.nextLine());
                while(t>0)
                {
                   String s = sc.nextLine();
                   Stack<Integer> st = new Stack<>();
                   for(int i=0;i<s.length();i++)
                   {
                      char c = s.charAt(i);
                      //If operand then push to stack
                      if(Character.isDigit(c)) //If b/w 0-9
                      {
                         st.push(c-'0'); //Character to Integer
                      }
                      else//Operator found
                      {
                         int right = st.pop();
                         int left = st.pop();
                       //  int res = calculate(left,right,c);
                         switch(c)
                         {
                            case '+':
                               st.push(left+right);
                               break;
                            case '-':
                               st.push(left-right);
                               break;
                            case '*':
                               st.push(left*right);
                               break;
                            case '/':
                               st.push(left/right);
                               break;
                         }
                       //  st.push(res);
                      }
                   }
                   System.out.println(st.pop());
                   t--;
                }
        }
// static int calculate(int left, int right, char c)
// {
//     if(c=='+') return left+right;
//     if(c=='-') return left-right;
//     if(c=='*') return left*right;
//     if(c=='/') return left/right;
//     return -1;
// }
}
```

→ 3 + 4 5 +

→ $\boxed{345*+}$

$\boxed{34*5+}$

fixed? → Order of operands
→ Only operators order changes

Order not Position

String res

List → LIFO(Stack)

1st 3+4*5   345*+

2nd 3*4+5   34*5+

If the precedence of operator on the top of stack is higher or equal then pop & add it to result.

3rd

2 + (3 * 4) − (5 ÷ 6)

res
234*+56÷−

Queue (FIFO)

→ Bank Center

→ Movie Ticket

→ CPU Disk Scheduling

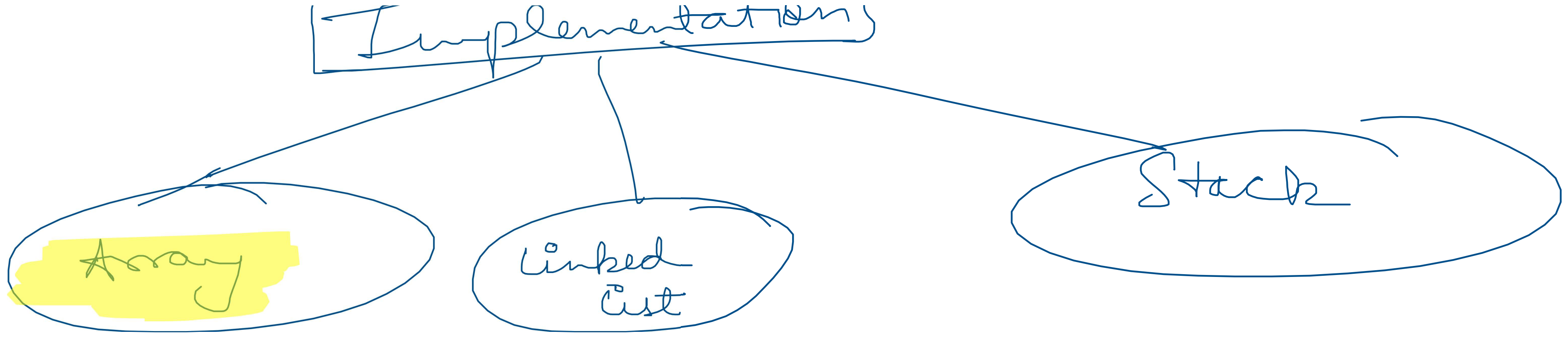
→ Printer

ADT → List of Operations

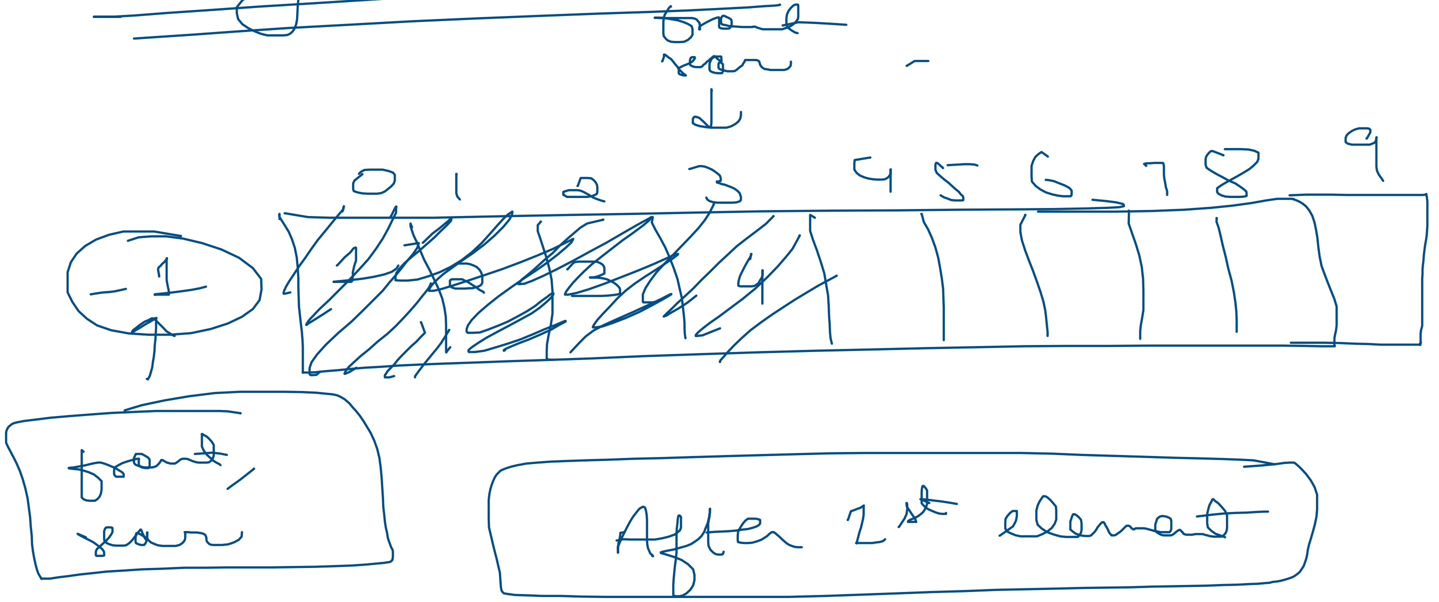
1. Insert → Enqueue
2. Remove → Dequeue
3. Peek ( )
4. isEmpty ( )

Queue ADT

→ O(1)



Array Implementation



Insertion / Enqueue happens at rear

Deletion / Dequeue happens at front