Infix to Postfix

```java
package stackInJava;

import java.util.Stack;

public class infixToPostfix {
    public static void main(String[] args) {
        String infix="9-(8*6/3)+8*2/2";  //CT_2

        Stack<String>val=new Stack<>();
        Stack<Character>op=new Stack<>();

        for(int i=0;i<infix.length();i++){
            char ch=infix.charAt(i);
            int ascii=(int)ch;

            if(ascii>=48 && ascii<=57){
                String s=""+ch;
                val.push(s);
            }
            else if(op.size()==0 || ch=='(' || op.peek()=='('){
                op.push(ch);
            }
            else if(ch==')'){
                while(op.peek()!='('){
                    String v2=val.pop();
                    String v1=val.pop();
                    char o=op.pop();
                    String t=v1+v2+o;
                    val.push(t);
                }
                op.pop();
            }
            else{
                if(ch=='+' || ch=='-'){
                    String v2=val.pop();
                    String v1=val.pop();
                    char o=op.pop();
                    String t=v1+v2+o;
                    val.push(t);

                    op.push(ch);
                }
                if(ch=='/' || ch=='*'){
```

```
                    if(op.peek()=='*' || op.peek()=='/'){
                        String v2=val.pop();
                        String v1=val.pop();
                        char o=op.pop();
                        String t=v1+v2+o;
                        val.push(t);

                        op.push(ch);
                    }
                    else op.push(ch);
                }
            }
        }
        while(val.size()>1){
            String v2=val.pop();
            String v1=val.pop();
            char o=op.pop();
            String t=v1+v2+o;
            val.push(t);
        }
        System.out.println(val.peek());
    }
}
```

postfix

```
package stackInJava;

import java.util.Stack;

public class postfix {
    public static void main(String[] args) {
        String str="953+4*6/-";
        Stack<Integer>val=new Stack<>();
        for(int i=0;i<str.length();i++){
            char ch=str.charAt(i);
            int ascii=(int)ch;
            if(ascii>=48 && ascii<=57){
                val.push(ascii-48);
            }
```

```
            else{
                int v2=val.pop();
                int v1=val.pop();
                if(ch=='+') val.push(v1+v2);
                if(ch=='-') val.push(v1-v2);
                if(ch=='/') val.push(v1/v2);
                if(ch=='*') val.push(v1*v2);
            }
        }
        System.out.println(val.peek());
    }
}
```

Quick Sort

```
#include <iostream>
using namespace std;

// Swap function
void swap(int &a, int &b) {
    int temp = a;
    a = b;
    b = temp;
}

// Partition function
int partition(int arr[], int low, int high) {
    int pivot = arr[high]; // পিভট ধরা হলো শেষ উপাদান
    int i = low - 1;

    for(int j = low; j < high; j++) {
        if(arr[j] < pivot) {
            i++;
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i + 1], arr[high]); // পিভটকে সঠিক জায়গায় বসানো
    return i + 1;
}

// Quick Sort function
void quickSort(int arr[], int low, int high) {
```

```
    if(low < high) {
        int pi = partition(arr, low, high); // পিভট ঠিক করা

        quickSort(arr, low, pi - 1);  // বাম পাশে sort
        quickSort(arr, pi + 1, high); // ডান পাশে sort
    }
}

int main() {
    int n;
    cout << "Enter number of elements: ";
    cin >> n;

    int arr[n];
    cout << "Enter elements: ";
    for(int i = 0; i < n; i++)
        cin >> arr[i];

    quickSort(arr, 0, n - 1);

    cout << "Sorted array: ";
    for(int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;

    return 0;
}
```

//Array Implements

```
package stackInJava;

public class arrayImplement {
    public static class Stack{
        int[] arr=new int[5];
        int idx=0;
        void push(int x){
            if(isFull()){
                System.out.println("Stack is full!");
                return;
            }
            arr[idx]=x;
```

```
            idx++;
        }
        int peek(){
            if(idx==0){
                System.out.println("Stack is Empty!");
                return -1;
            }
            return arr[idx-1];
        }
        int pop(){
            if(idx==0){
                System.out.println("Stack is Empty!");
                return -1;
            }
            int top=arr[idx-1];
            arr[idx-1]=0;
            idx--;
            return top;
        }
        void display(){
            for(int i=0;i<idx-1;i++){
                System.out.println(arr[i]+" ");
            }
            System.out.println();
        }
        int size(){
            return idx;
        }
        boolean isEmpty(){
            if(idx==0) return true;
            else return false;
        }
        boolean isFull(){
            if(idx==arr.length) return true;
            else return false;
        }
    }
    public static void main(String[] args) {
        Stack st=new Stack();
        st.push(1);
        st.push(2);
        st.push(3);
        st.push(4);
        st.display();
        st.pop();
```

```
        st.display();
        st.push(3);
        System.out.println("size is: "+st.size());

        st.display();
    }
}
```

LinkedList Implement

```
package stackInJava;

public class linkedListImplement {

    // Make Stack class static so it can be used from main
    static class Stack {
        // Node class for linked list
        private class Node {
            int data;
            Node next;

            Node(int data) {
                this.data = data;
                this.next = null;
            }
        }

        // Top of the stack
        private Node top;

        // Constructor
        public Stack() {
            top = null;
        }

        // Push operation
        public void push(int value) {
            Node newNode = new Node(value);
            newNode.next = top;
            top = newNode;
        }
```

```
        // Pop operation
        public int pop() {
            if (isEmpty()) {
                throw new RuntimeException("Stack Underflow - Cannot pop from an
empty stack");
            }
            int poppedValue = top.data;
            top = top.next;
            return poppedValue;
        }

        // Peek operation
        public int peek() {
            if (isEmpty()) {
                throw new RuntimeException("Stack is empty - Cannot peek");
            }
            return top.data;
        }

        // Check if the stack is empty
        public boolean isEmpty() {
            return top == null;
        }

        // Display stack (for testing)
        public void display() {
            Node temp = top;
            System.out.print("Stack: ");
            while (temp != null) {
                System.out.print(temp.data + " ");
                temp = temp.next;
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        Stack stack = new Stack();

        stack.push(10);
        stack.push(20);
        stack.push(30);
        stack.display();  // Output: Stack: 30 20 10

        System.out.println("Top element is: " + stack.peek());  // Output: 30
```

```
        System.out.println("Popped element: " + stack.pop());  // Output: 30
        stack.display();  // Output: Stack: 20 10

        System.out.println("Is stack empty? " + stack.isEmpty());  // Output:
false
    }
}
```