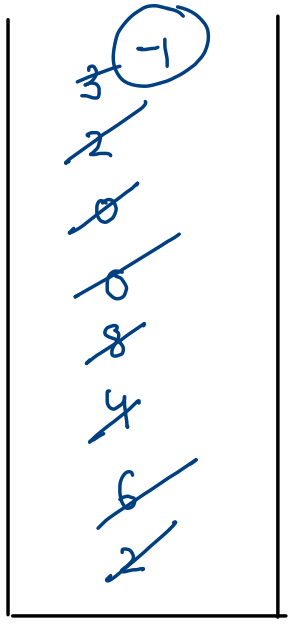
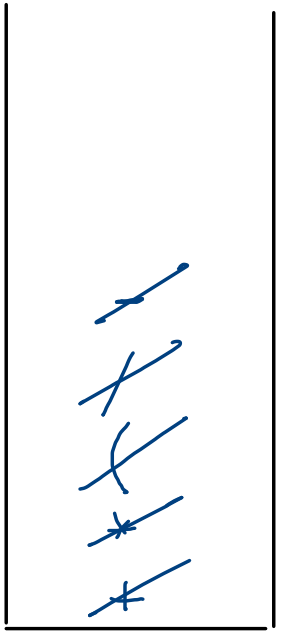
Infix Evaluation

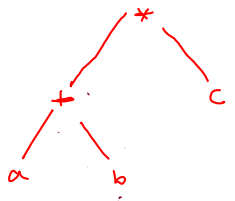
③ ② ① ④

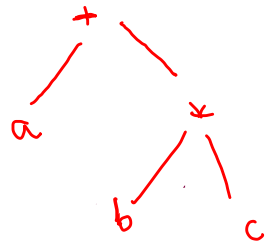
2 + 6 *(4 / 8)- 3





exp	infix	prefix	postfix
(a + b) * c	a + b * c	* + ab c	a b + c *
a + (b * c)	a + b * c	+ a * b c	a b c * +



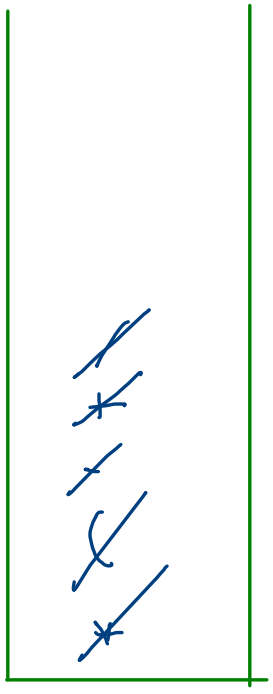


Infix Conversions

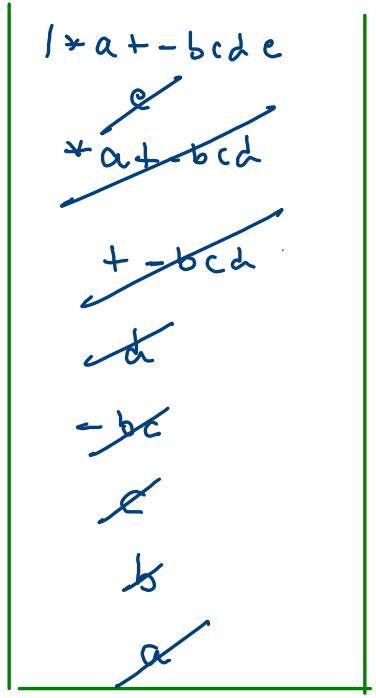
a*(b-c+d)/e

abc-d+*e/

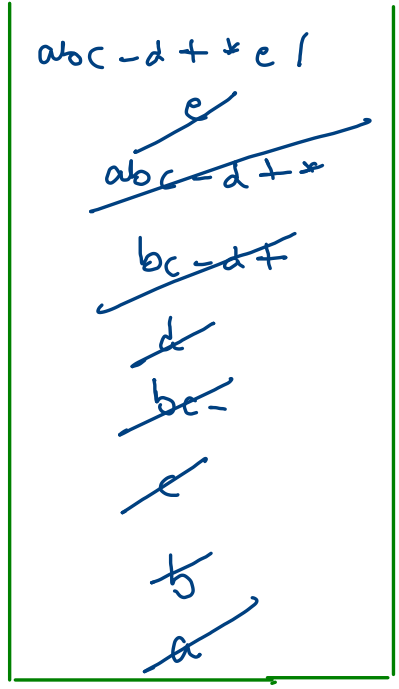
/*a+-bcde



opr st



prefix



postfix

opr = /

pre-> / | *a+-bcd | e

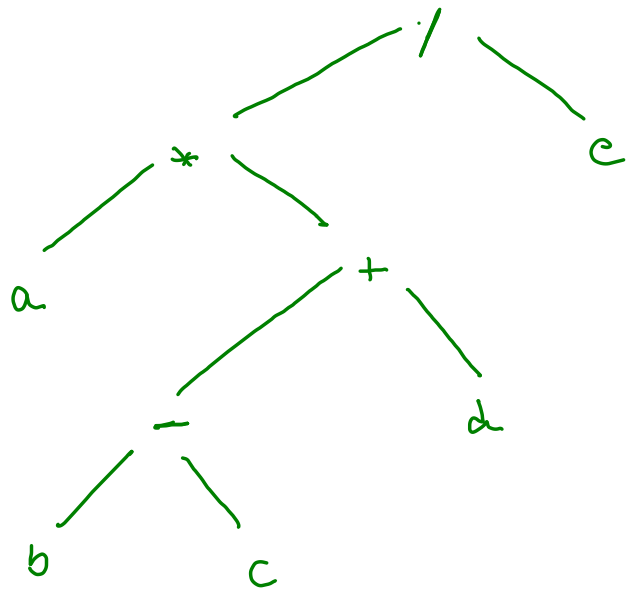
opr | av | av

post-> abc-d+* | e | /

av | av | opr

③ ① ② ④

a*(b-c+d)/e



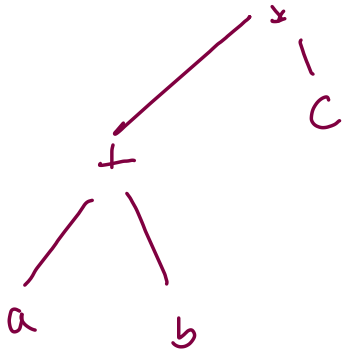
④ ③ ② ①

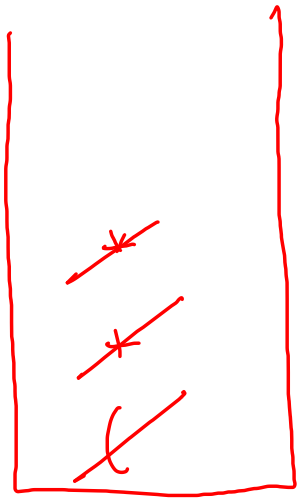
/ * a + - b c d e

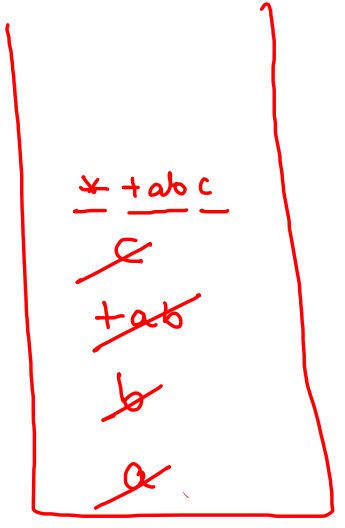
① ② ③ ④

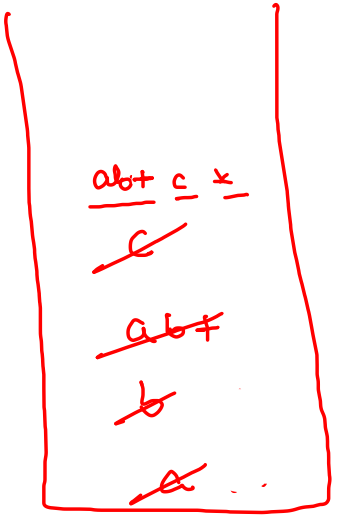
a b c - d + * e /

(a + b) * c







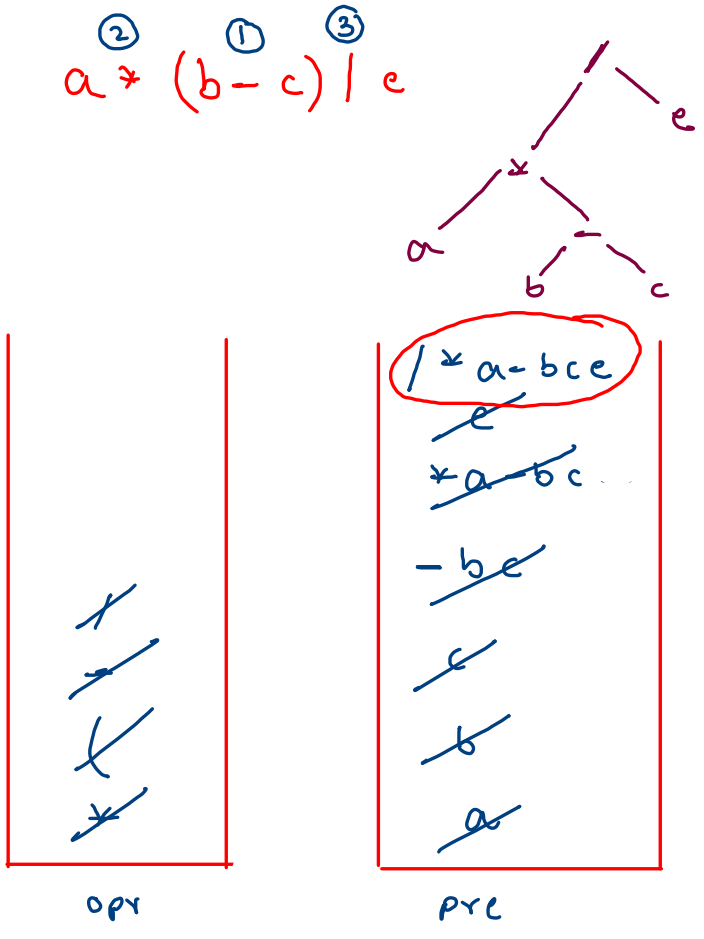


pre -> opr lv rv

post -> lv rv opr

```
else if((ch >= '0' && ch <= '9') || (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')) {
    pre.push(ch + "");
    post.push(ch + "");
}
else if(ch == '+' || ch == '-' || ch == '*' || ch == '/') {
    while(oprst.size() > 0 && oprst.peek() != '(' && priority(oprst.peek()) >= priority(ch)) {
        //evaluate
        char opr = oprst.pop();

        //work in pre stack
        String prerv = pre.pop();
        String prelv = pre.pop();

        String prev = opr + prelv + prerv;
        pre.push(prev);

        //work in post stack
        String porv = post.pop();
        String polv = post.pop();

        String pov = polv + porv + opr;
        post.push(pov);
    }

    oprst.push(ch);
}
```

```
if(ch == '(') {
    oprst.push(ch);
}
else if(ch == ')') {
    //evaluate till an opening

    while(oprst.peek() != '(') {
        char opr = oprst.pop();

        //work in pre stack
        String prerv = pre.pop();
        String prelv = pre.pop();

        String prev = opr + prelv + prerv;
        pre.push(prev);

        //work in post stack
        String porv = post.pop();
        String polv = post.pop();

        String pov = polv + porv + opr;
        post.push(pov);
    }
    oprst.pop();
}
```



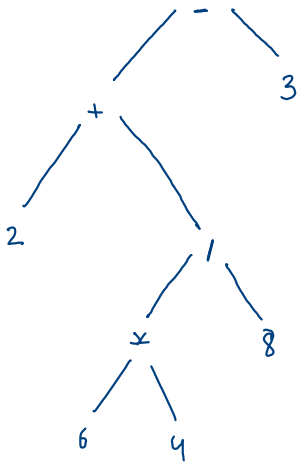
Postfix Evaluation And Conversions

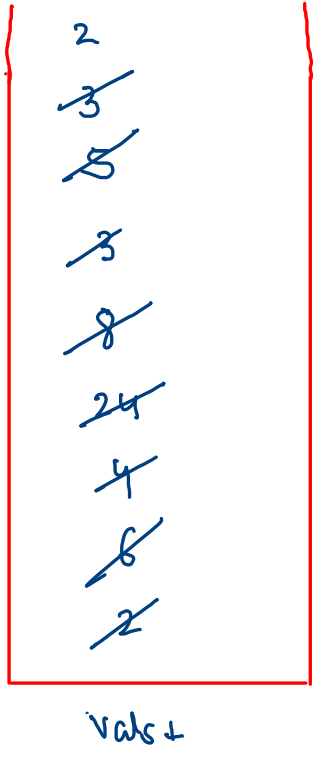
264*8/+3-

2
((2+((6*4)/8))-3)
-+2/*6483

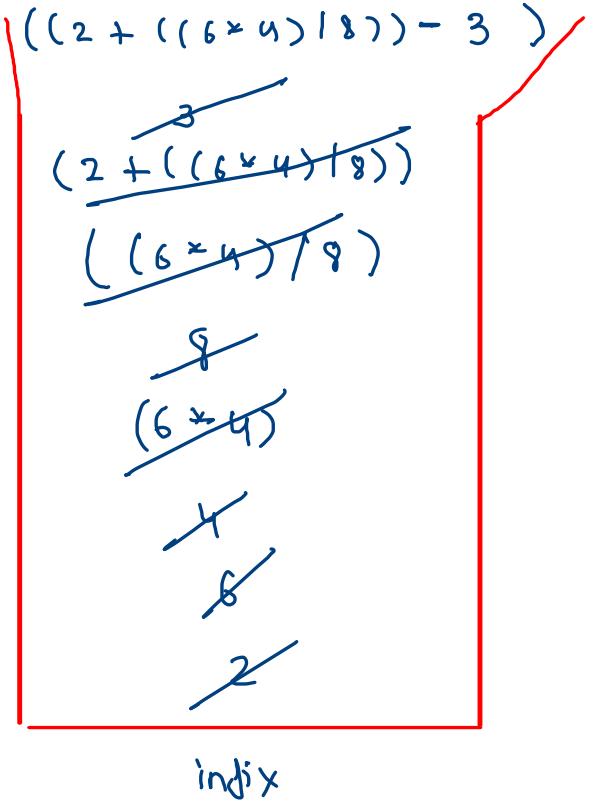
in-> (lv opr rv)

pre-> opr lv rv

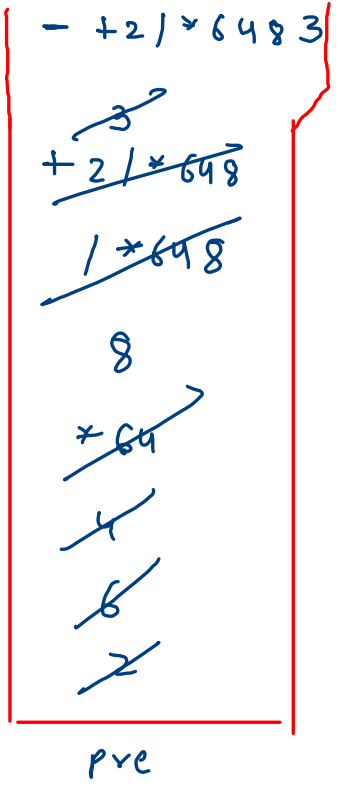




postfix



infix

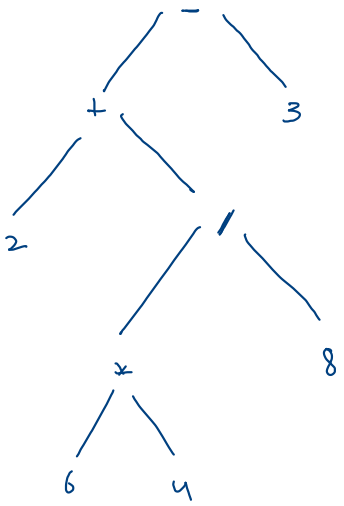


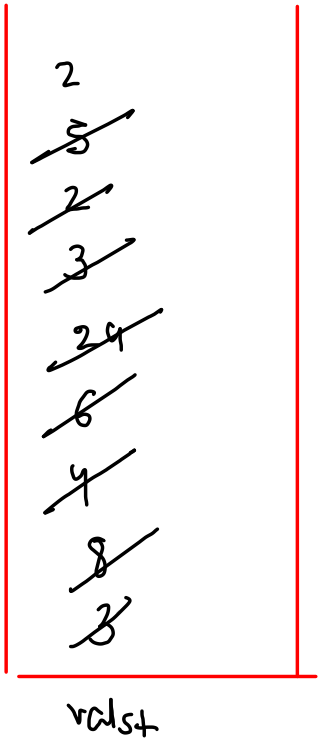
pre

Prefix Evaluation Ar
Conversions

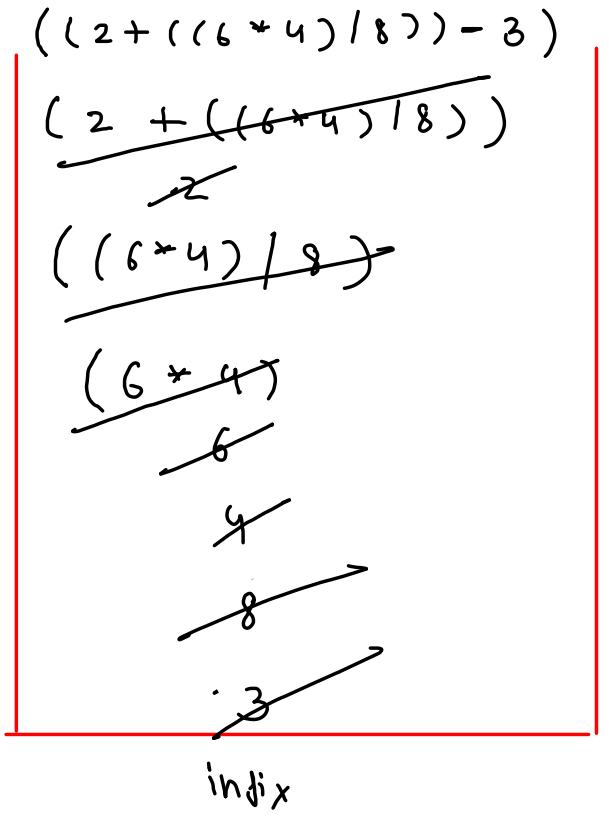
-+2/*6483

right to left

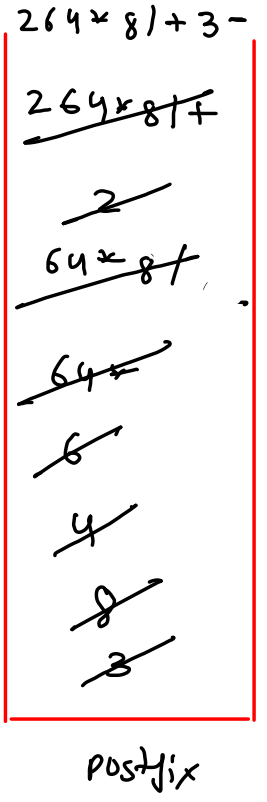




postfix

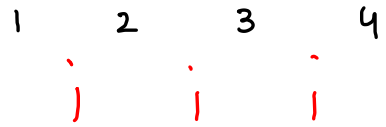


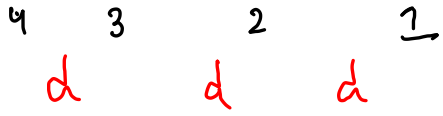
infix



postfix

Smallest Number Following Pattern





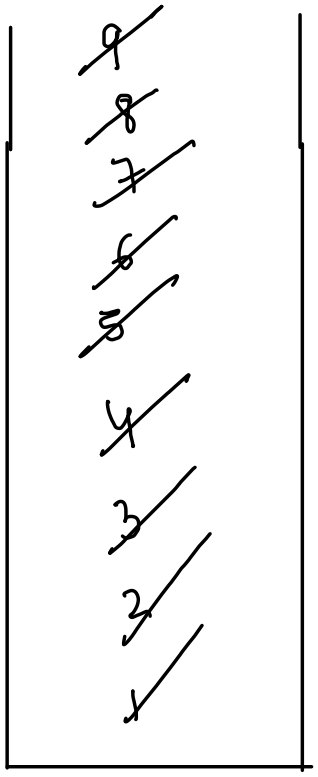
d -> 21
i -> 12
ddd -> 4321
iii -> 1234
dddiddd -> 43218765
iiddd -> 126543

rules :

(i) pattern.length() ≤ 8

(ii) ans : distinct digits (1 to 9)

(iii) ans.length -> pat.length()+1;



4 3 2 1 5 8 7 6 9

d d d i i d d i

↑

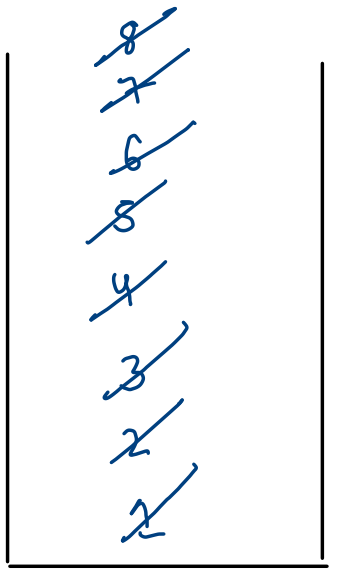
c = ~~0~~ 1
2
3
4
5
6
7
8
9

```
if (ch == 'd') {
        c++;
        st.push(c);
}
else {
        c++;
        st.push(c);
        print(st)
}
```

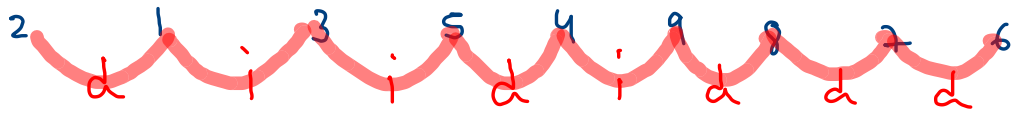
4 3 2 1 5 7 6 8

d d d i i d i

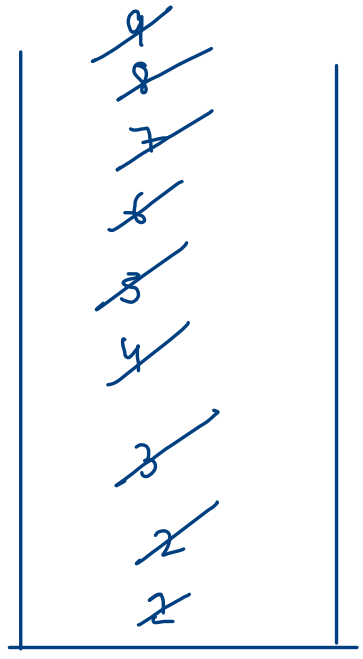
```
for(int i=0; i < str.length();i++) {
    char ch = str.charAt(i);


    if(ch == 'd') {
        c++;
        st.push(c);
    }
    else if(ch == 'i') {
        c++;
        st.push(c);

        //print the stack
        while(st.size() > 0) {
            System.out.print(st.pop());
        }
    }
}

c++;
st.push(c);

//print the stack
while(st.size() > 0) {
    System.out.print(st.pop());
}
```

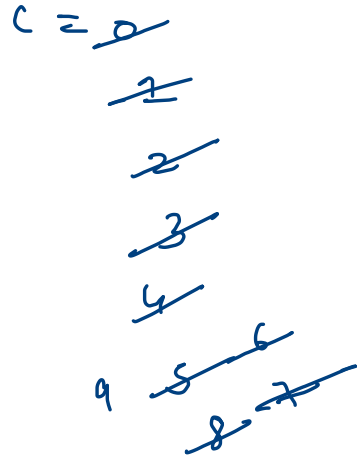


c = 0 1 2 3 4 5 6 7 8



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


    if(ch == 'd') {
        c++;
        st.push(c);
    }
    else if(ch == 'i') {
        c++;
        st.push(c);

        //print the stack
        while(st.size() > 0) {
            System.out.print(st.pop());
        }
    }
}

c++;
st.push(c);

//print the stack
while(st.size() > 0) {
    System.out.print(st.pop());
}
```

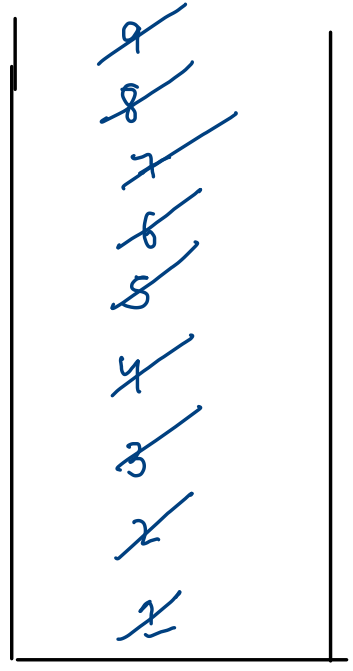




1 2 6 5 4 3 9 8 7

i i d d d i d d .

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

    if(ch == 'd') {
        c++;
        st.push(c);
    }
    else if(ch == 'i') {
        c++;
        st.push(c);

        //print the stack
        while(st.size() > 0) {
            System.out.print(st.pop());
        }
    }
}

c++;
st.push(c);

//print the stack
while(st.size() > 0) {
    System.out.print(st.pop());
}
```



c = 0 1 2 3 4 6 5 7 8 9