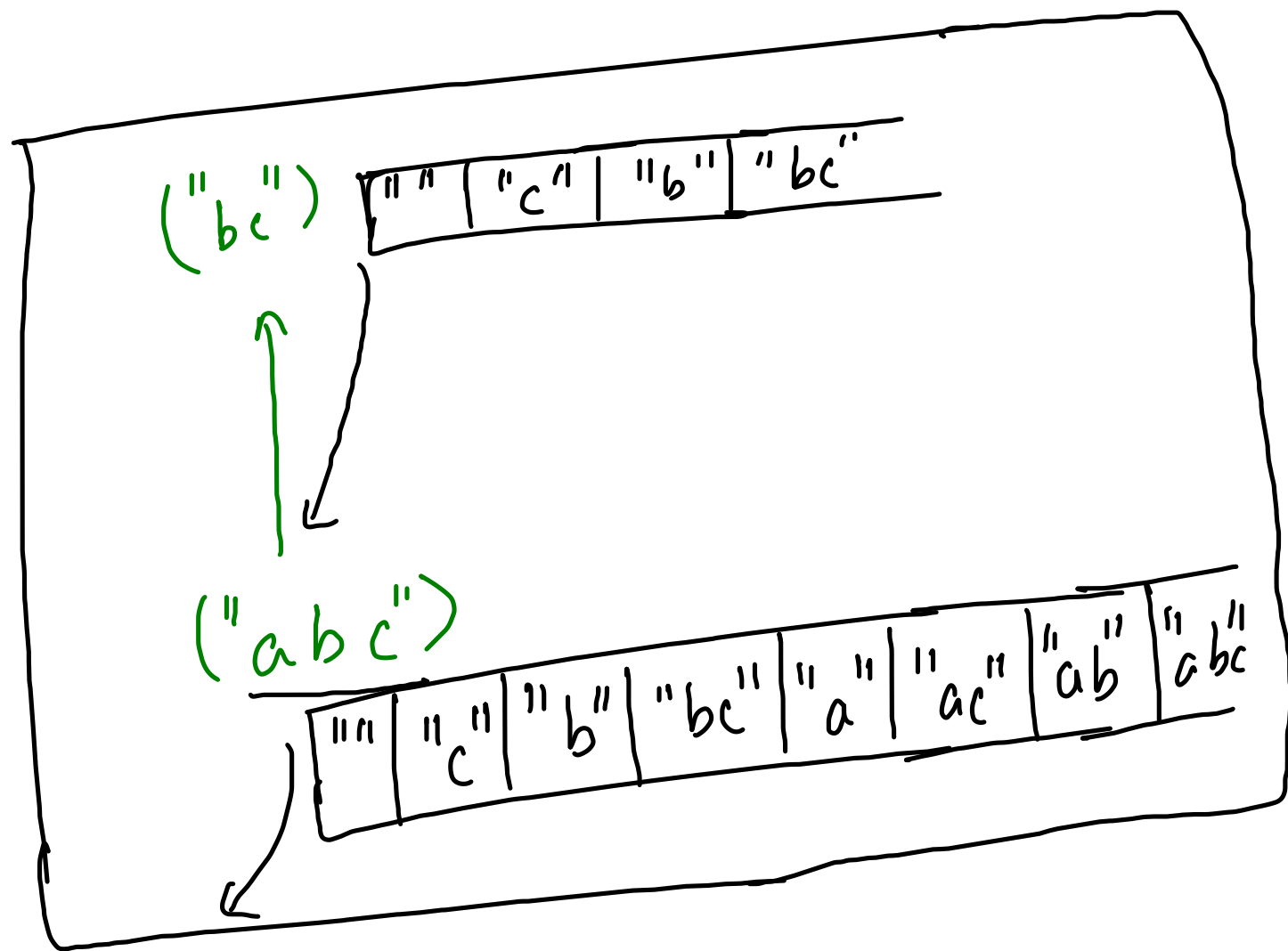


Subsequence \rightarrow Each & every character has binary choice (0 \rightarrow not inc) & (1 \rightarrow inc)

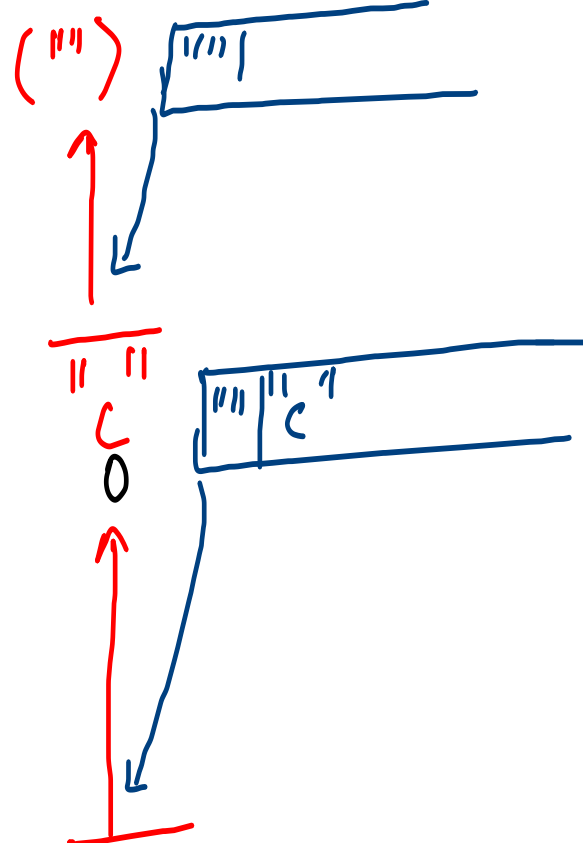
What

"abc"
000 \rightarrow - - - \rightarrow ""
001 \rightarrow - - c \rightarrow "c"
010 \rightarrow - b - \rightarrow "b"
011 \rightarrow - b c \rightarrow "bc"
100 \rightarrow a - - \rightarrow "a"
101 \rightarrow a - c \rightarrow "ac"
110 \rightarrow a b - \rightarrow "ab"
111 \rightarrow a b c \rightarrow "abc"

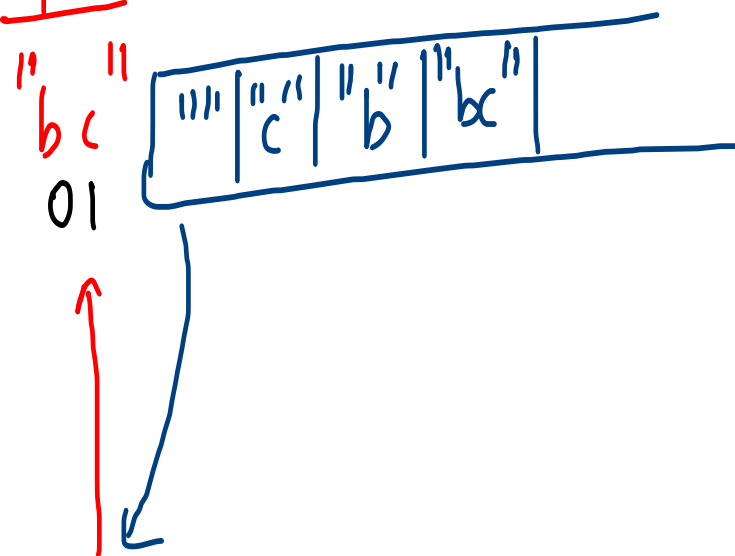
"bc"
00 \rightarrow ""
01 = "c"
10 = "b"
11 = "bc"



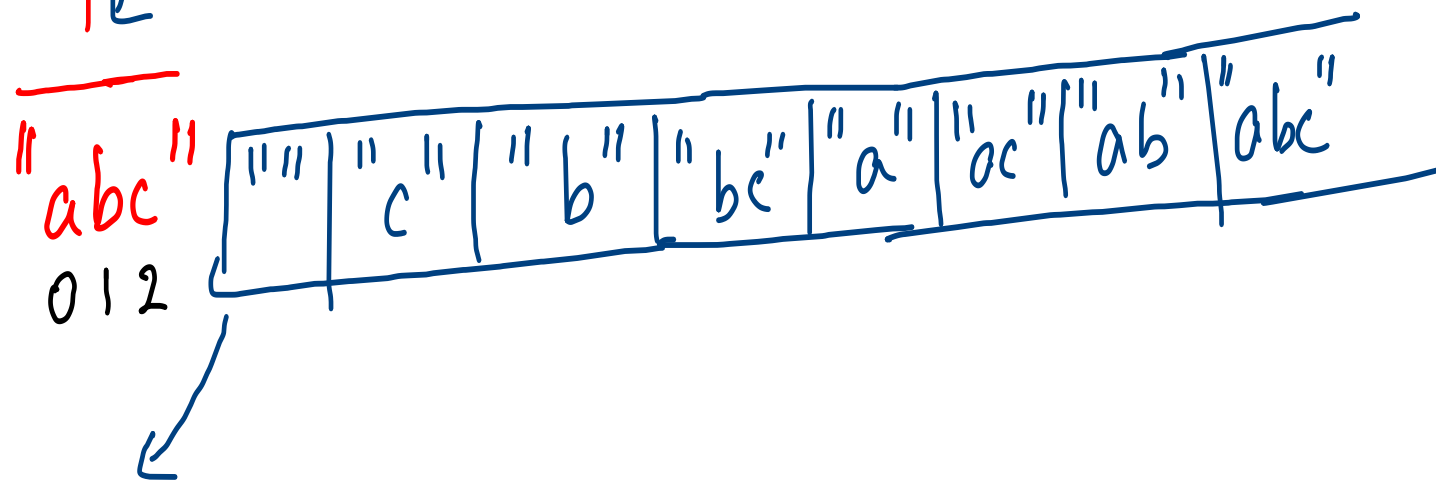
ch = 'c'
sos = ""



ch = 'b'
sos = 'c'

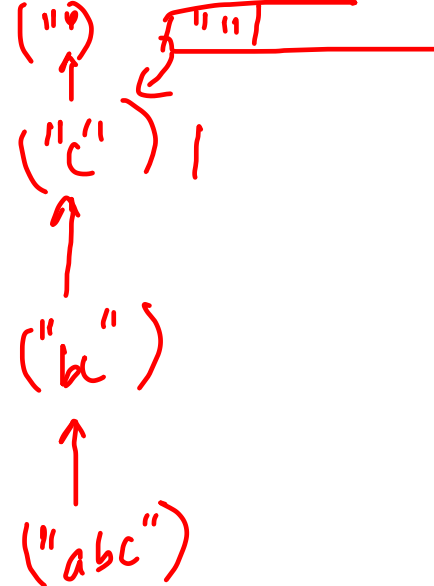


ch = 'a'
sos = 'bc'



[char ch ← str.charAt(0); ✓
 String sos ← str.substring(1); ✓]

[ArrayList]



```

public static ArrayList<String> gss(String str) {
    if(str.length() == 0){
        ArrayList<String> base = new ArrayList<>();
        base.add("");
        return base;
    }

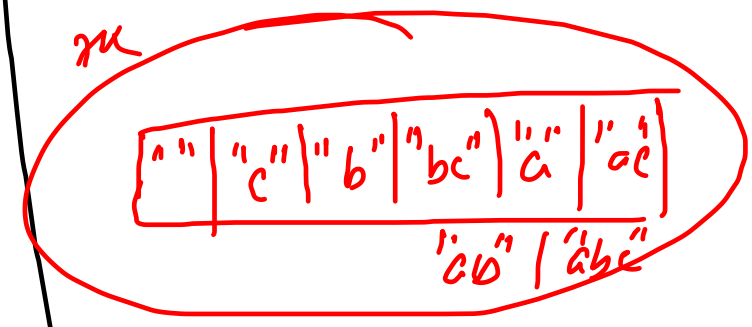
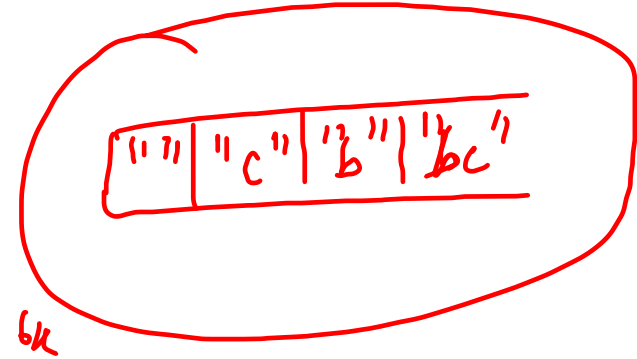
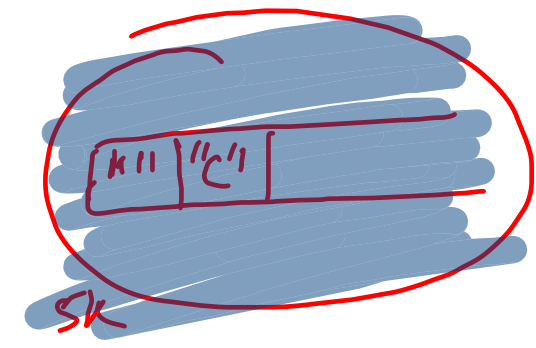
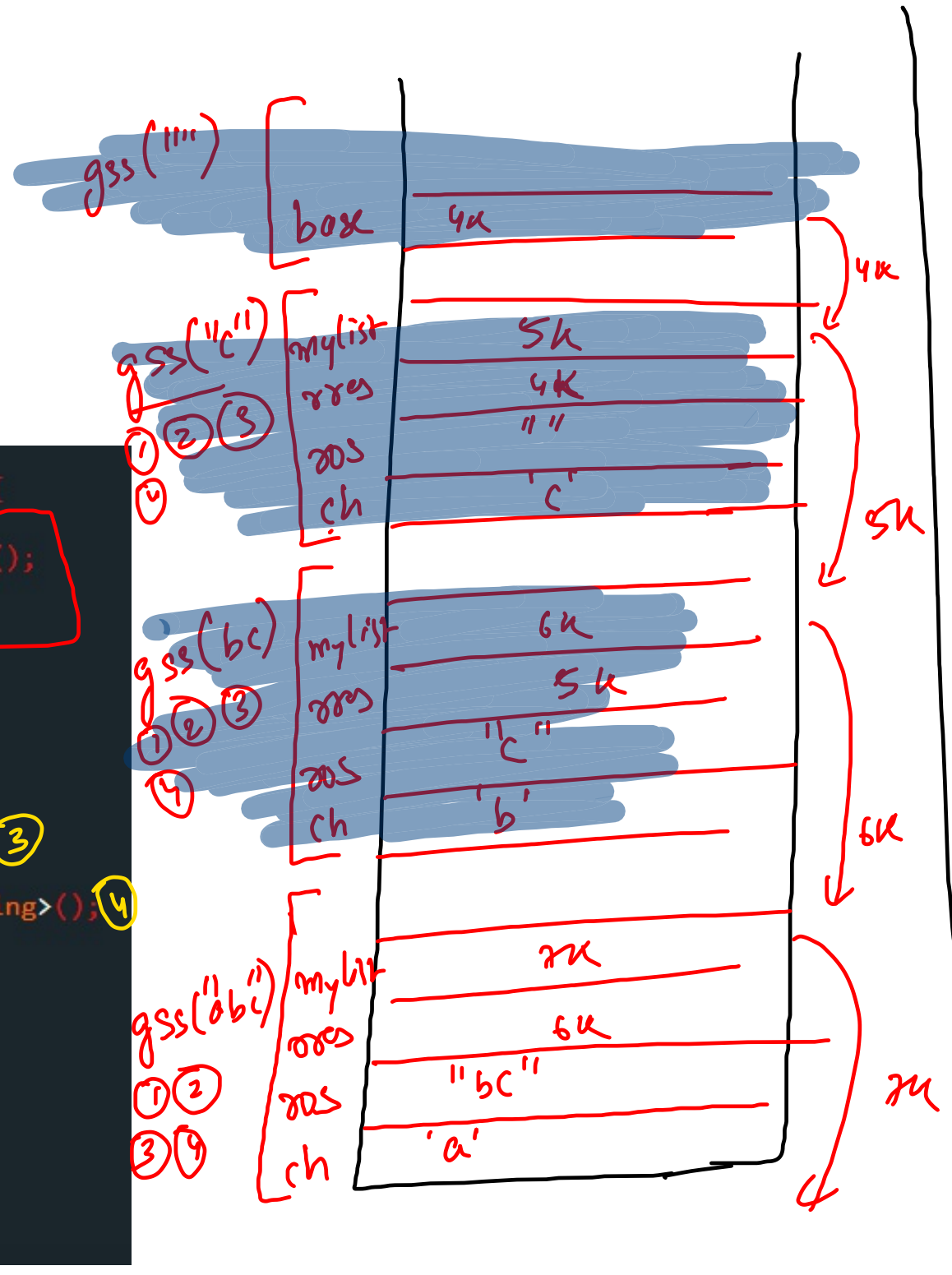
    char ch = str.charAt(0);
    String ros = str.substring(1);

    ArrayList<String> rres = gss(ros); // faith

    ArrayList<String> myList = new ArrayList<String>();
    for(String s : rres){
        myList.add(s);
    }
    for(String s : rres){
        myList.add(ch + s);
    }

    return myList;
}

```

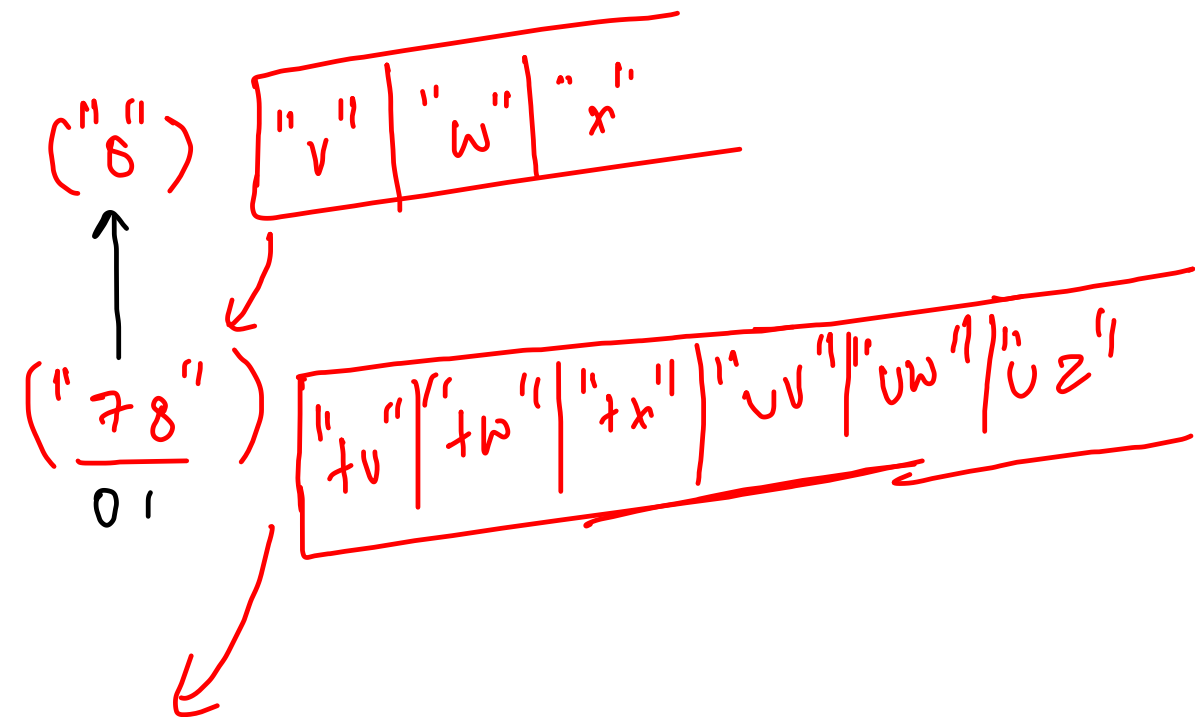


```

public static String keypad(char ch){
    if(ch == '1') return "abc";
    else if(ch == '2') return "def";
    else if(ch == '3') return "ghi";
    else if(ch == '4') return "jkl";
    else if(ch == '5') return "mno";
    else if(ch == '6') return "pqrs";
    else if(ch == '7') return "tu";
    else if(ch == '8') return "vwx";
    else if(ch == '9') return "yz";
    else return ".;";
}

```

$ch = 'a'$
 $ans = "8"$
 $mod = "tu"$



```

public static String keypad(char ch){
    if(ch == '1') return "abc";
    else if(ch == '2') return "def";
    else if(ch == '3') return "ghi";
    else if(ch == '4') return "jkl";
    else if(ch == '5') return "mno";
    else if(ch == '6') return "pqrs";
    else if(ch == '7') return "tu";
    else if(ch == '8') return "vwx";
    else if(ch == '9') return "yz";
    else return ".;";
}

```

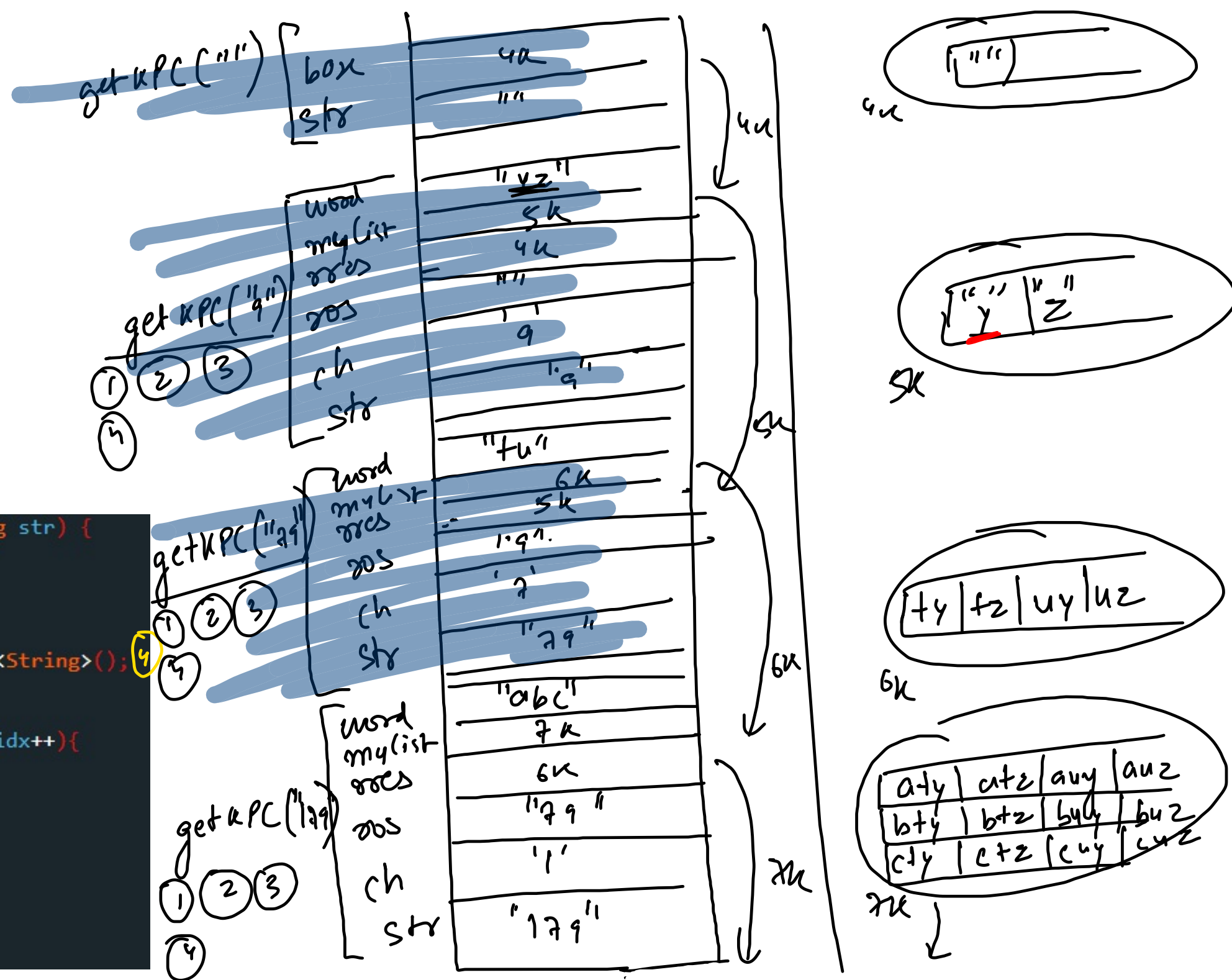
```

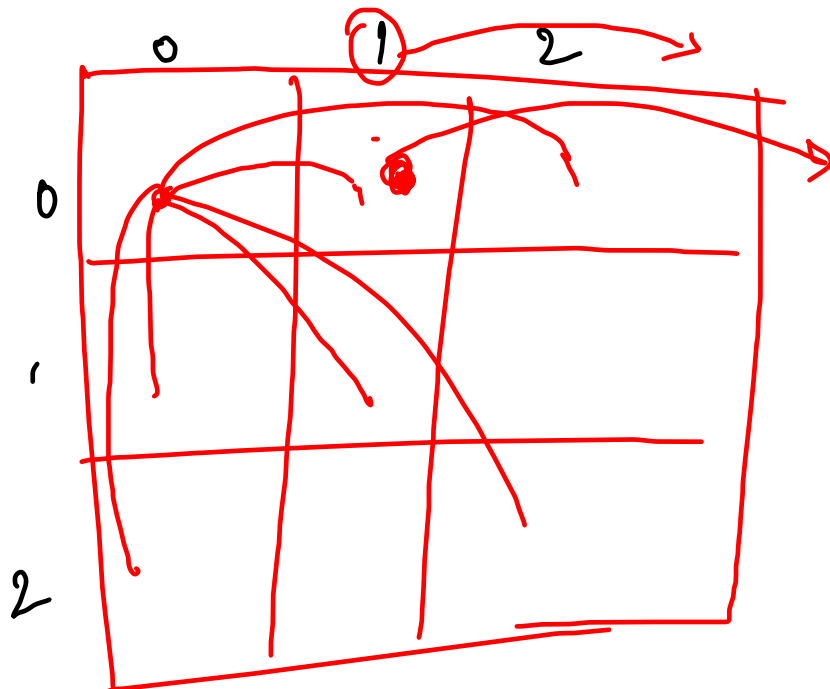
public static ArrayList<String> getKPC(String str) {
    ① char ch = str.charAt(0);
    ② String ros = str.substring(1);
    ArrayList<String> rres = getKPC(ros); ③
    ArrayList<String> myList = new ArrayList<String>(); ④

    String word = keypad(ch);
    for(int idx = 0 ; idx < word.length() ; idx++){
        String opt = word.charAt(idx);
        for(String poss : rres){
            myList.add(opt+poss);
        }
    }

    return myList;
}

```

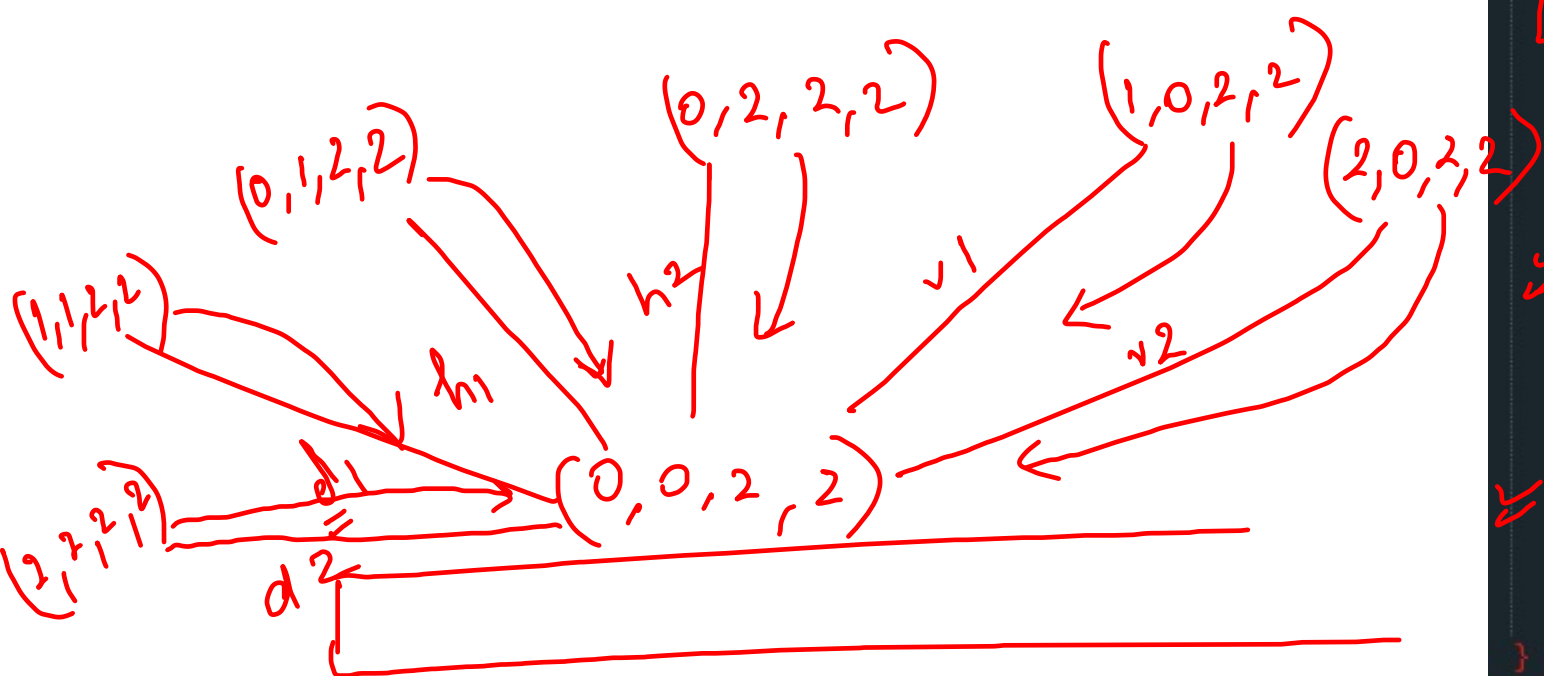




nr = 3

nc = 3

jmp = 2



```
public static ArrayList<String> getMazePaths(int sr, int sc, int dr, int dc) {
    if(sr == dr && sc == dc){
        ArrayList<String> base = new ArrayList<String>();
        base.add("");
        return base;
    }
    ArrayList<String> myPath = new ArrayList<>();

    // Horizontal
    for(int jmp = 1; sc+jmp <= dc; jmp++){
        ArrayList<String> rres = getMazePaths(sr, sc+jmp, dr, dc);
        for(String path : rres){
            myPath.add("h"+jmp+path);
        }
    }

    // vert.
    for(int jmp = 1; sr+jmp <= dr; jmp++){
        ArrayList<String> rres = getMazePaths(sr+jmp, sc, dr, dc);
        for(String path : rres){
            myPath.add("v"+jmp+path);
        }
    }

    // diag
    for(int jmp = 1; sr+jmp <= dr && sc+jmp <= dc; jmp++){
        ArrayList<String> rres = getMazePaths(sr+jmp, sc+jmp, dr, dc);
        for(String path : rres){
            myPath.add("d"+jmp+path);
        }
    }

    return myPath;
}
```

str = " h e l l o "

0 1 2 3 4

length() = 5

substring(idx): "This fun. is used to extract part of the string."

str.substring(0) = "hello"

" = (1) = "ello"

" = (2) = "llo"

" = (3) = "lo"

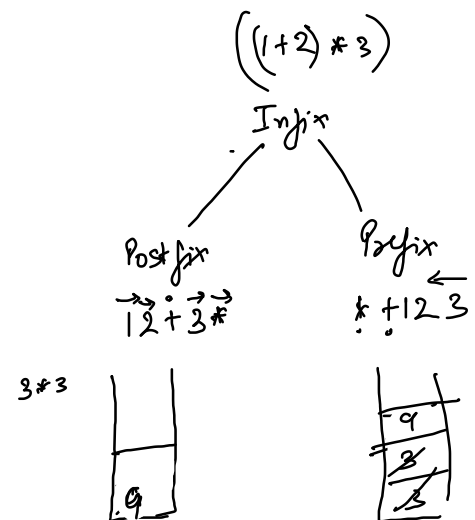
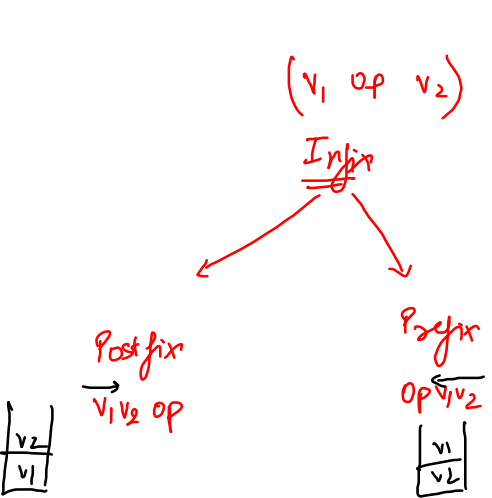
" = (4) = "o"

str.substring(5) = ""

Special case

str.substring(idx):

[0 → length]



Infix

$(((((10 + 2) * 5) / 6) + 3))$

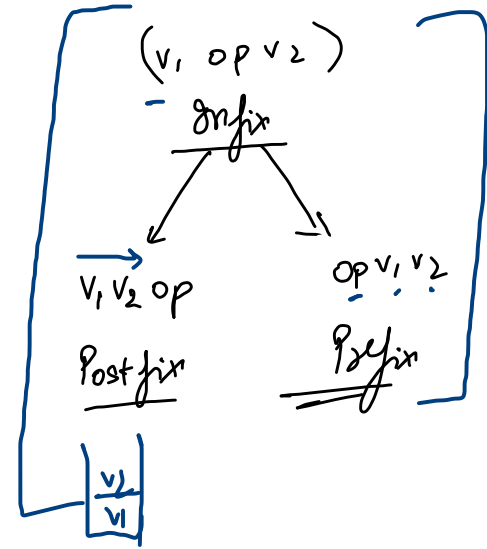
Postfix

Prefix

Ambiguous Exp

\rightarrow brackets

\rightarrow precedence

$$2 \cdot ((2 + ((6 \cdot 4) / 8)) - 3) + 2 \cdot 6483$$

$$\phi = 1$$

✓	✓	✓	•	✓	✓	✓	✓	•
0	1	2	3	4	5	6	7	8
2	6	4	*	8	/	+	3	-

5 - 3 * 2 ✓

~~3~~

~~2 + 3 = 5~~

~~24 / 8 = 3~~

~~8~~

~~6 * 4 = 24~~

~~4~~

~~6~~

~~2~~

v. 6

$$\left| \frac{((2 + ((6 \times 4) / 8)) - 3)}{3} \right|_{r_2}$$

$$4 * 8 / + 3 -$$

$((6 \times 4) / 8)$
8
$(6 * 4)$
4
6
2

Input (String)

$$\begin{array}{r} - +2/x6483 \\ \hline \text{" " " } \\ 3 \\ \hline \end{array}$$

$$\begin{array}{r} +2/\cancel{*648} \\ \hline \end{array}$$

$$\begin{array}{r} \cancel{/ *648} \\ \hline \text{" " " } \\ 8 \\ \hline \end{array}$$

$$\begin{array}{r} \text{" " " } \\ \cancel{*64} \\ \hline \end{array}$$

$$\begin{array}{r} \text{" " " } \\ 4 \\ \hline \end{array}$$

$$\begin{array}{r} \text{" " " } \\ 6 \\ \hline \end{array}$$

$$\begin{array}{r} \text{" " " } \\ 2 \\ \hline \end{array}$$

v_2
 v_1

Prefix <String>

✓ 2 ✓ 6 ✓ 4 ✓ * 8 / + 3 -

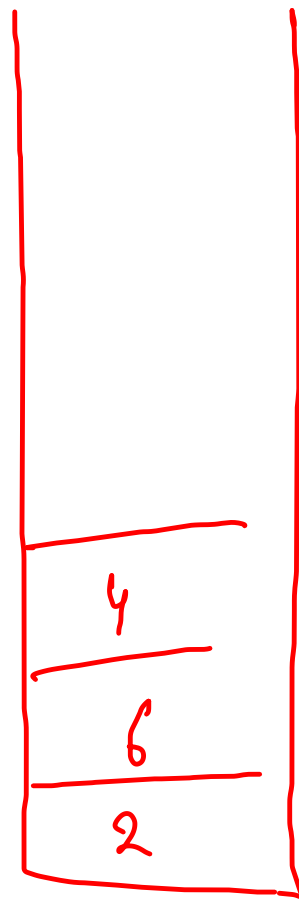
```
public static void solve(String exp){
    Stack<Integer> eval = new Stack<Integer>();
    Stack<String> preExp = new Stack<>();
    Stack<String> inExp = new Stack<>();

    for(int idx = 0 ; idx < exp.length() ; idx++){
        char ch = exp.charAt(idx);

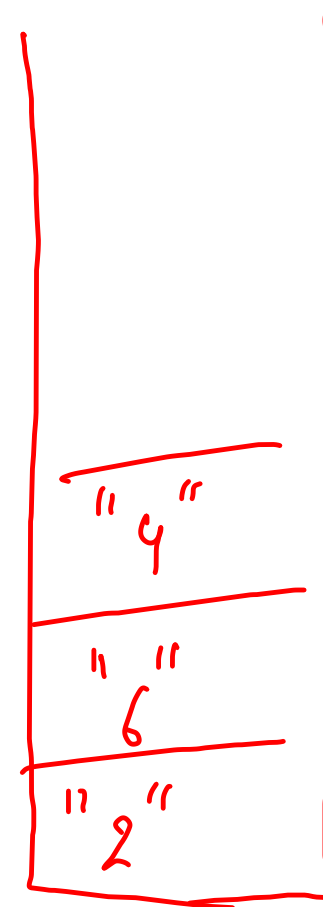
        if(ch == '+' || ch == '*' || ch == '/' || ch == '-'){
            evaluation(eval,ch);
            prefixExpBuild(preExp,ch);
            infixExpBuild(inExp,ch);
        }else{
            eval.push(Integer.parseInt(ch+""));
            inExp.push(ch+"");
            preExp.push(ch+"");
        }
    }

    System.out.println(eval.pop());
    System.out.println(inExp.pop());
    System.out.println(preExp.pop());
}

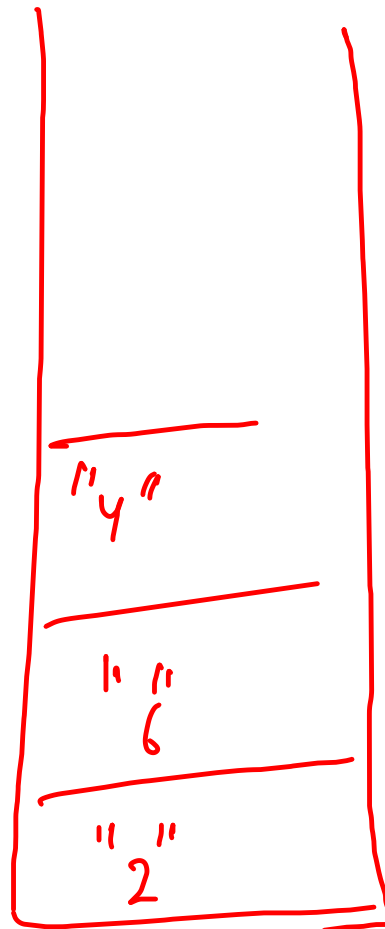
public static void evaluation(Stack<Integer> eval , char op){}
public static void prefixExpBuild(Stack<String> preExp , char op){}
public static void infixExpBuild(Stack<String> inExp,char op){}
```



Eval



PreExp



InExp

Sample Input

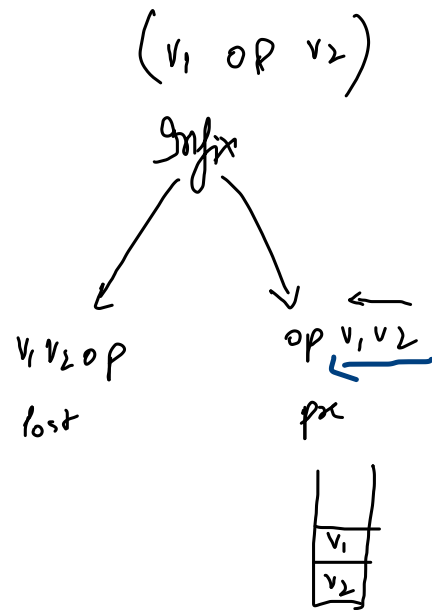
$-+2/*6483$

Sample Output

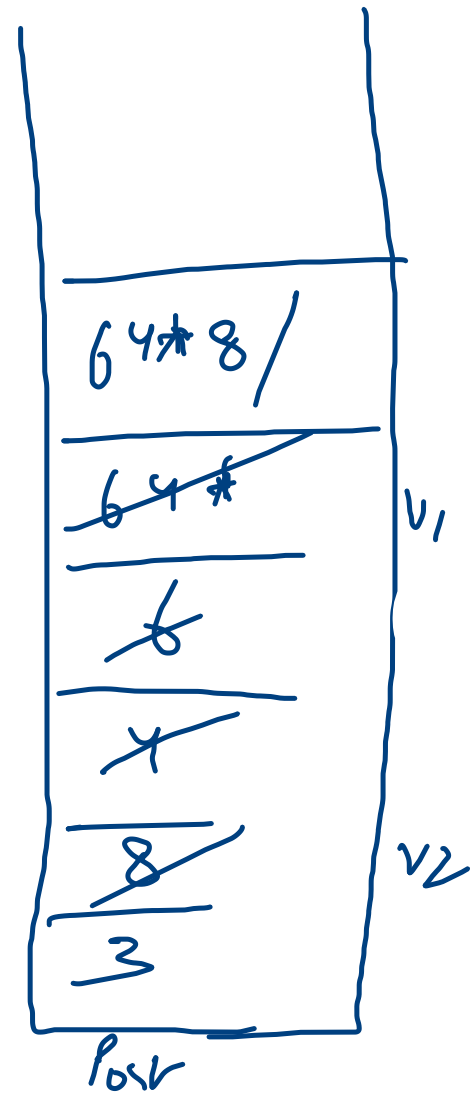
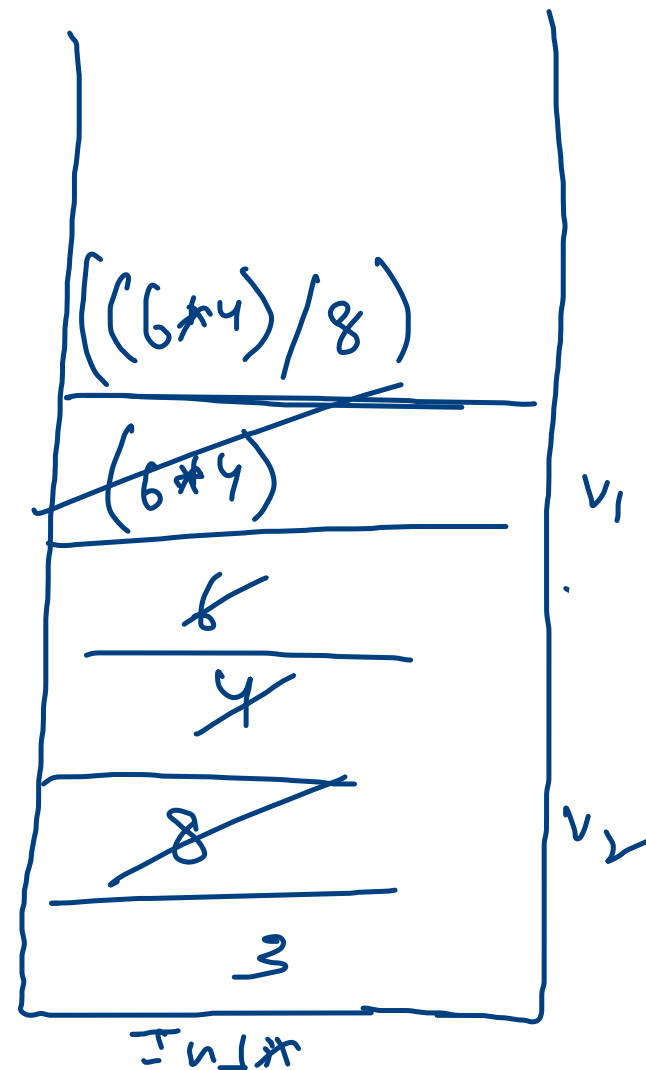
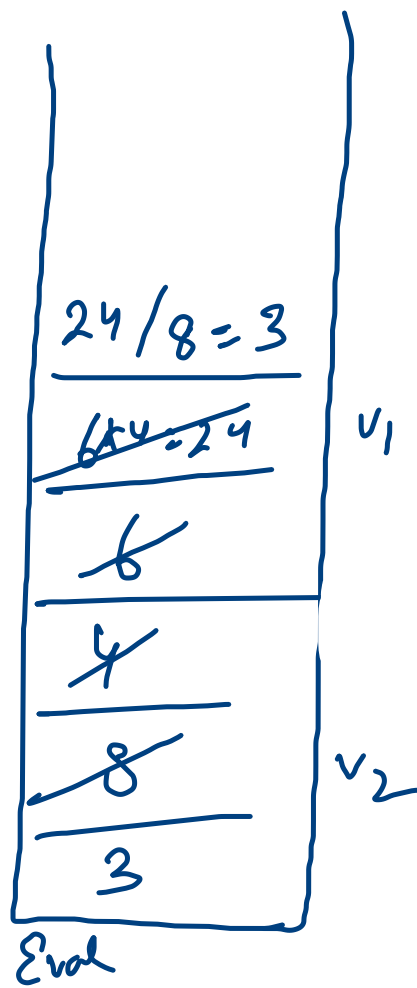
2

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

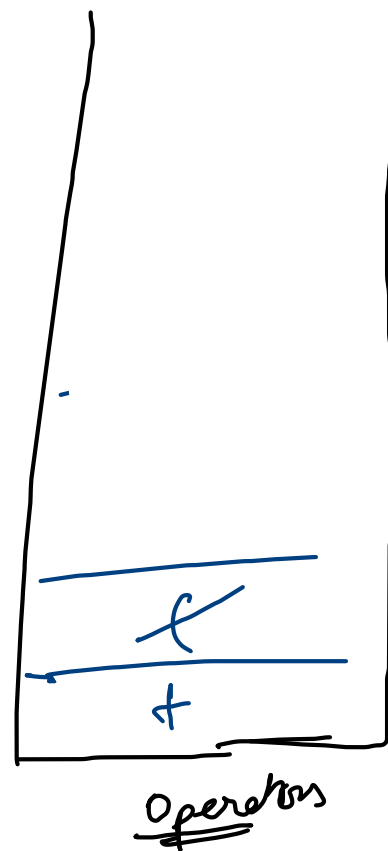
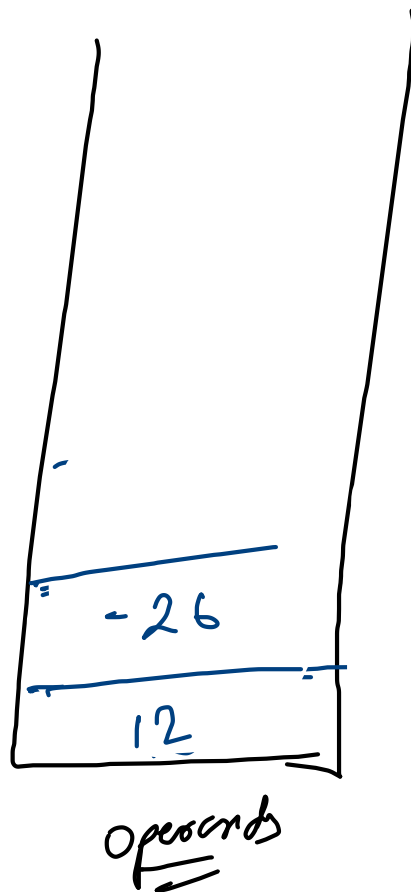
$264*8/+3-$



Handwritten expression: $-+2/*6483$ with checkmarks above the numbers and a circle around the asterisk.



$$\left((8/2) * 2 \right) + 4 + (6-8 * 4) / 8$$

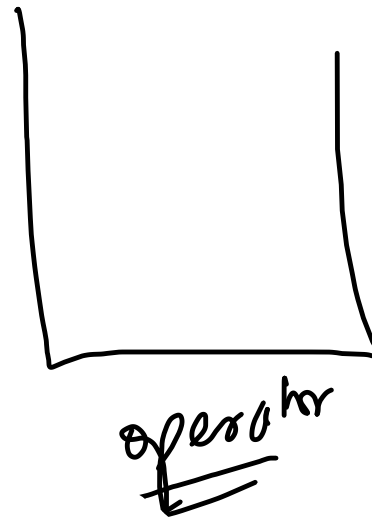
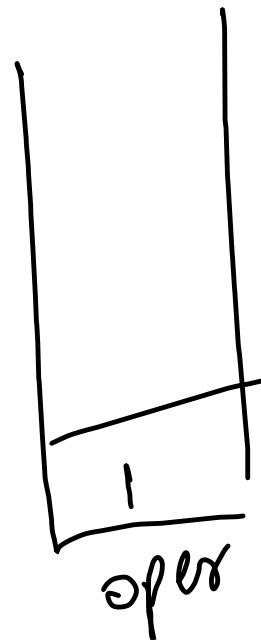


op =

st. peek =

$$\underline{\text{precedence}(\text{st. peek}()) \geq \text{precedence}(\text{op})}$$

✓
✓
1+2



```
while(operators.size() > 0 && operators.peek() != '(' && precedence(operators.peek()) >= precedence(ch)){
```

