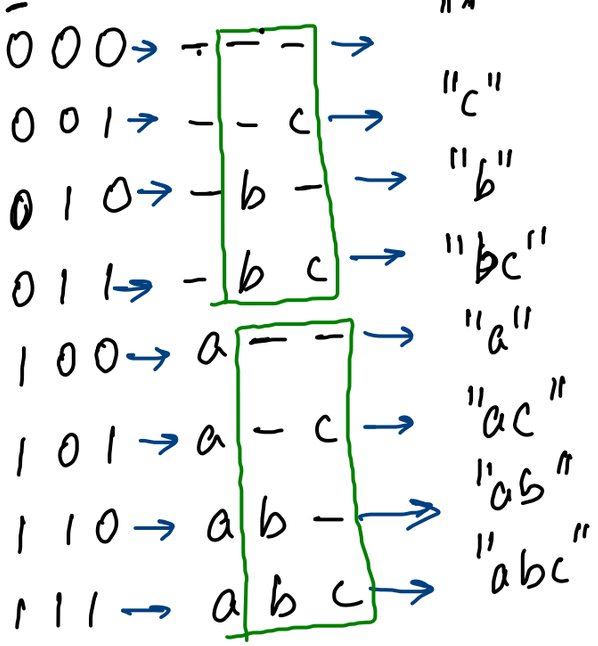
Subsequence → Each & every character has binary choice (0 → not inc) & (1 - inc)

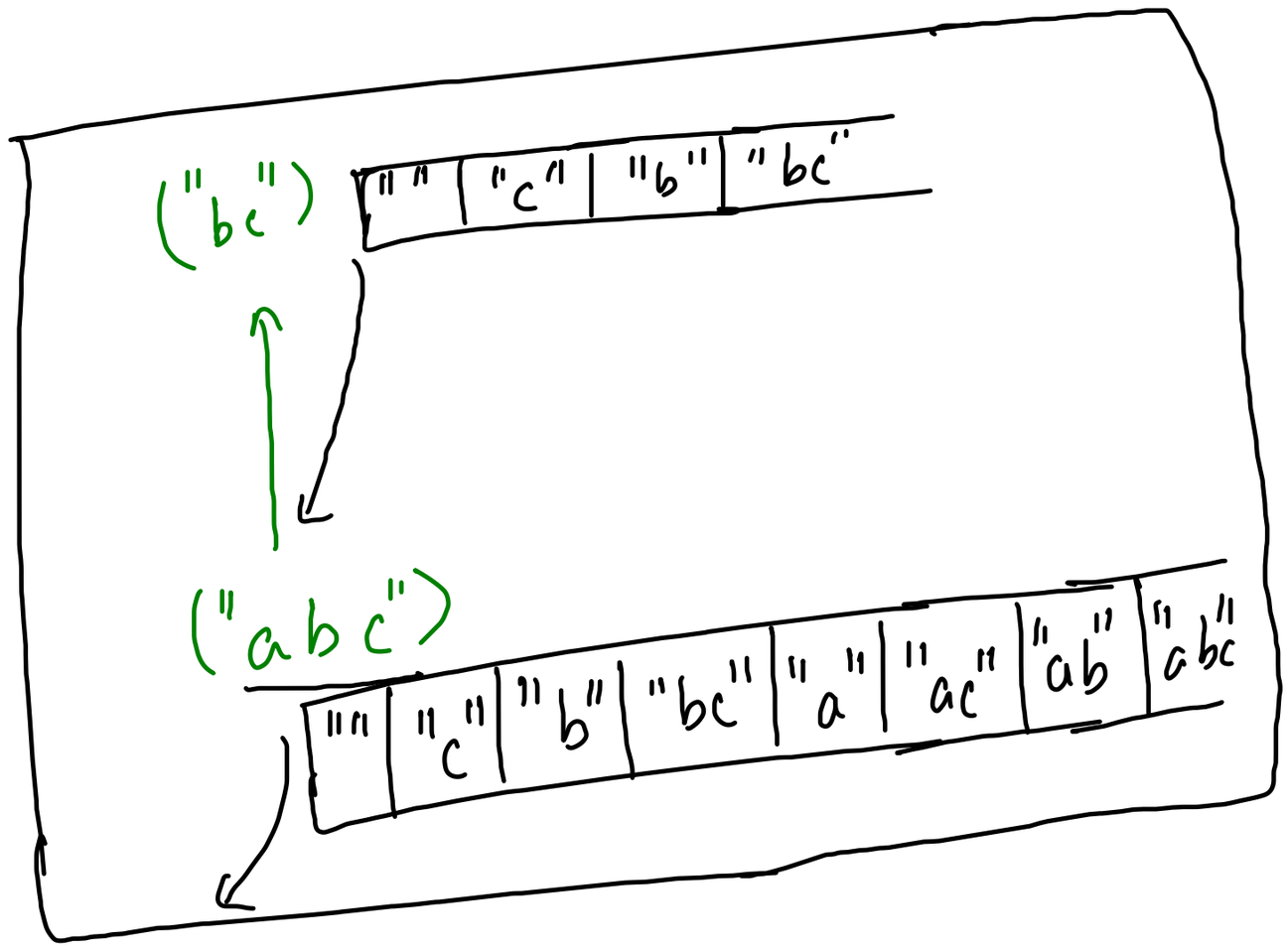
"abc"

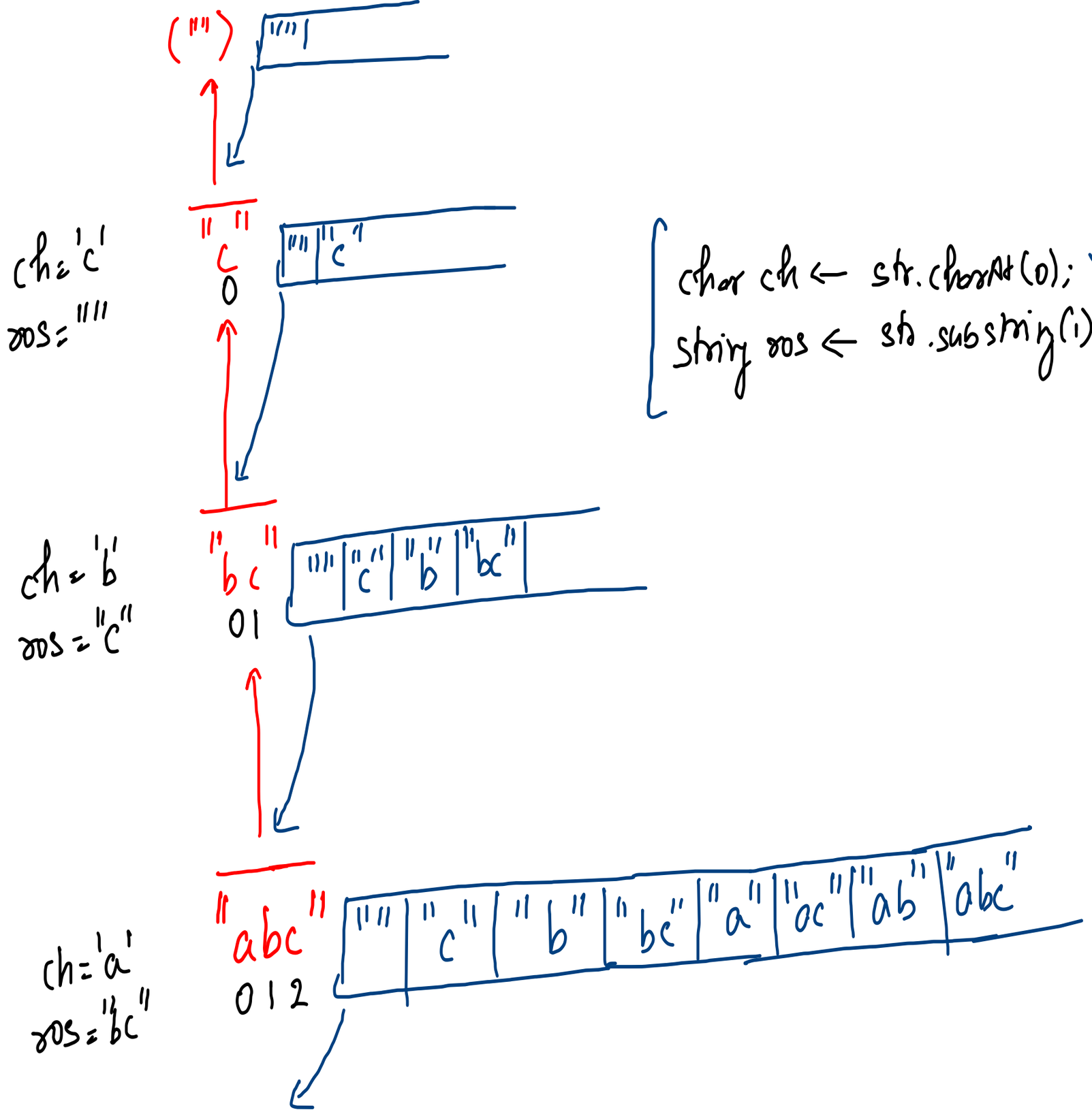


"bc"

00 → ""
01 = "c"
10 = "b"
11 = "bc"

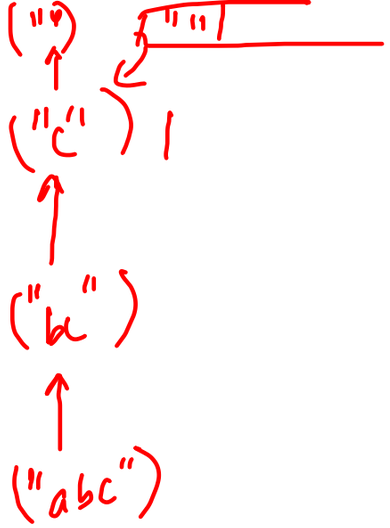
What



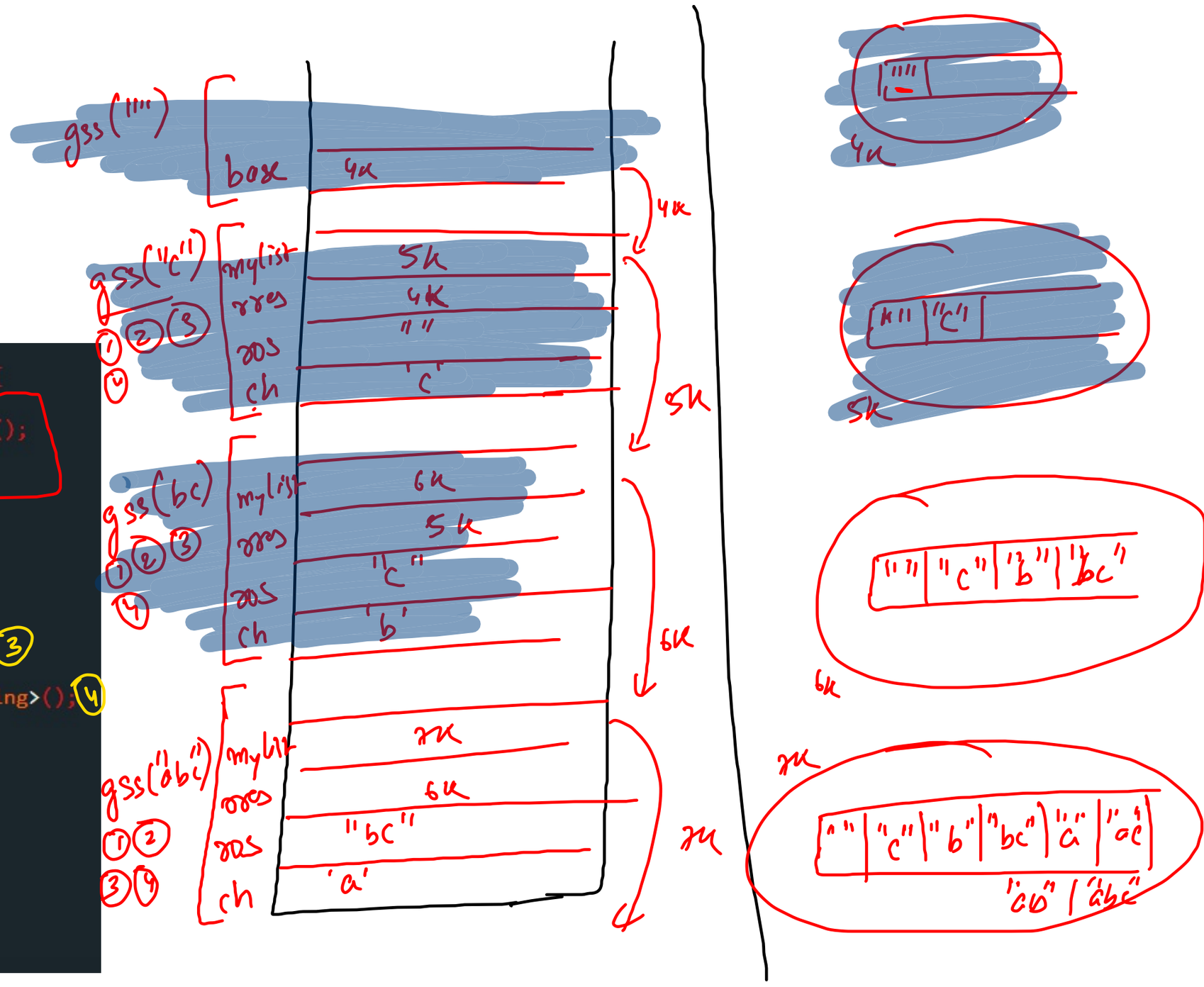



```
char ch ← str.charAt(0);
String ros ← str.substring(1);
```

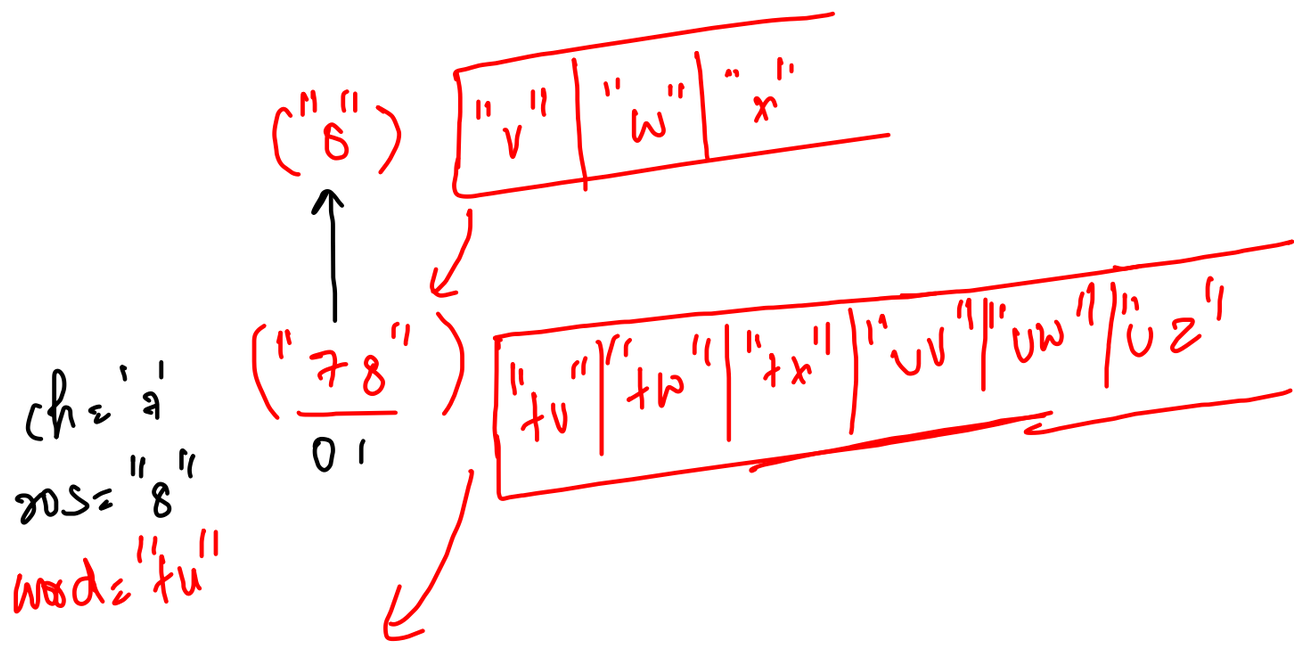
ArrayList




```java
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;
}
```

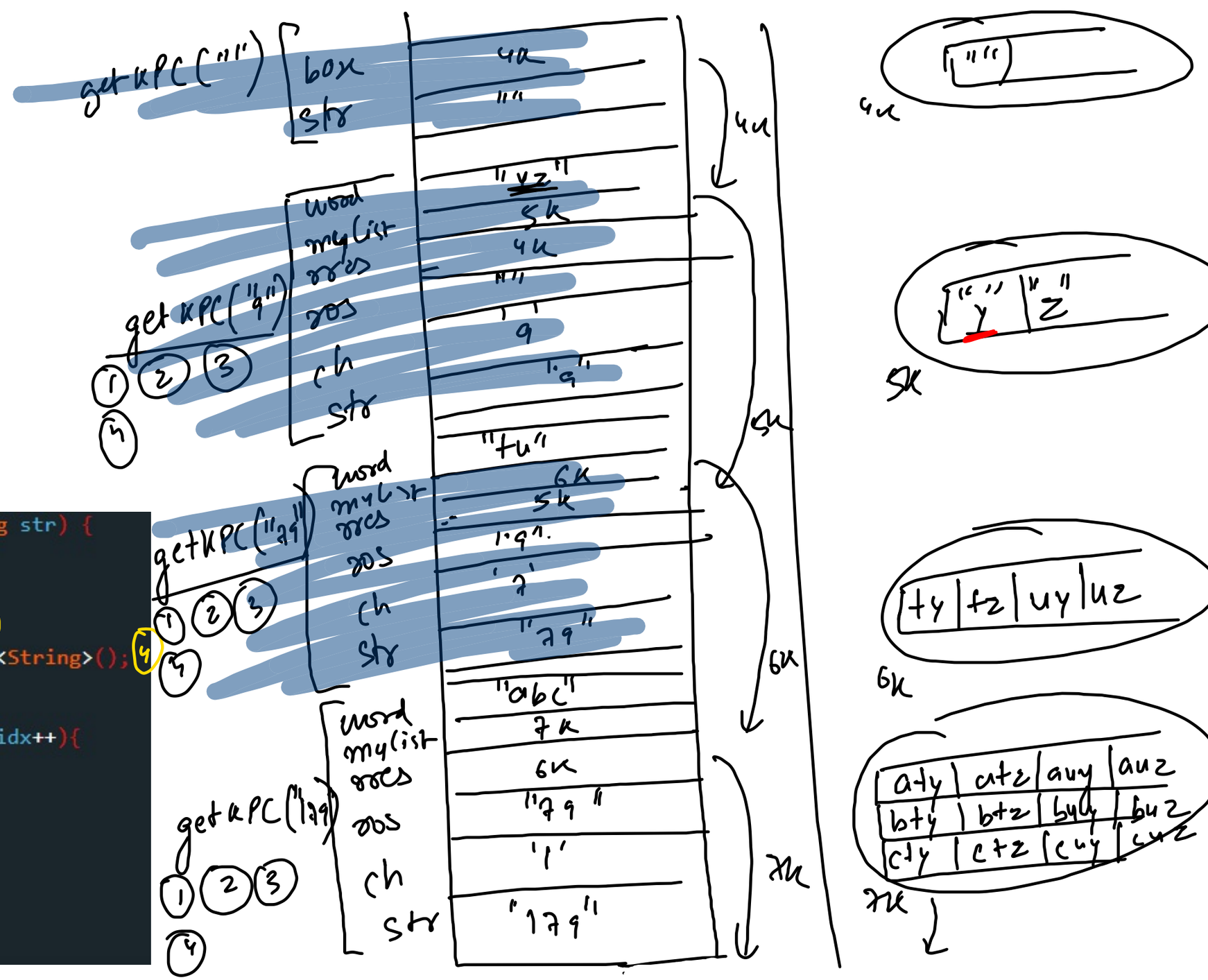


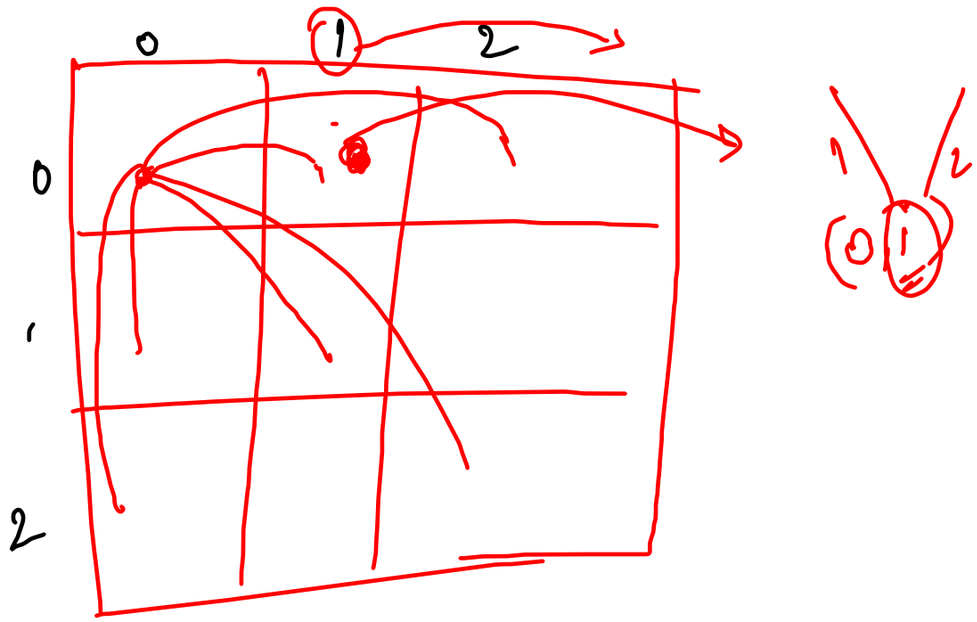
```java
public static String keypad(char ch){
    if(ch == '1') return "abc";
    else if(ch == '2') return "def";
    else if(ch == '3') return "ghi";
    else if(ch == '4') return "jkl";
    else if(ch == '5') return "mno";
    else if(ch == '6') return "pqrs";
    else if(ch == '7') return "tu";
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



```java
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 ".;";
}
```

```java
public static ArrayList<String> getKPC(String str) {

    char ch = str.charAt(0);
    String ros = str.substring(1);
    ArrayList<String> rres = getKPC(ros);
    ArrayList<String> myList = new ArrayList<String>();

    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;
}
```



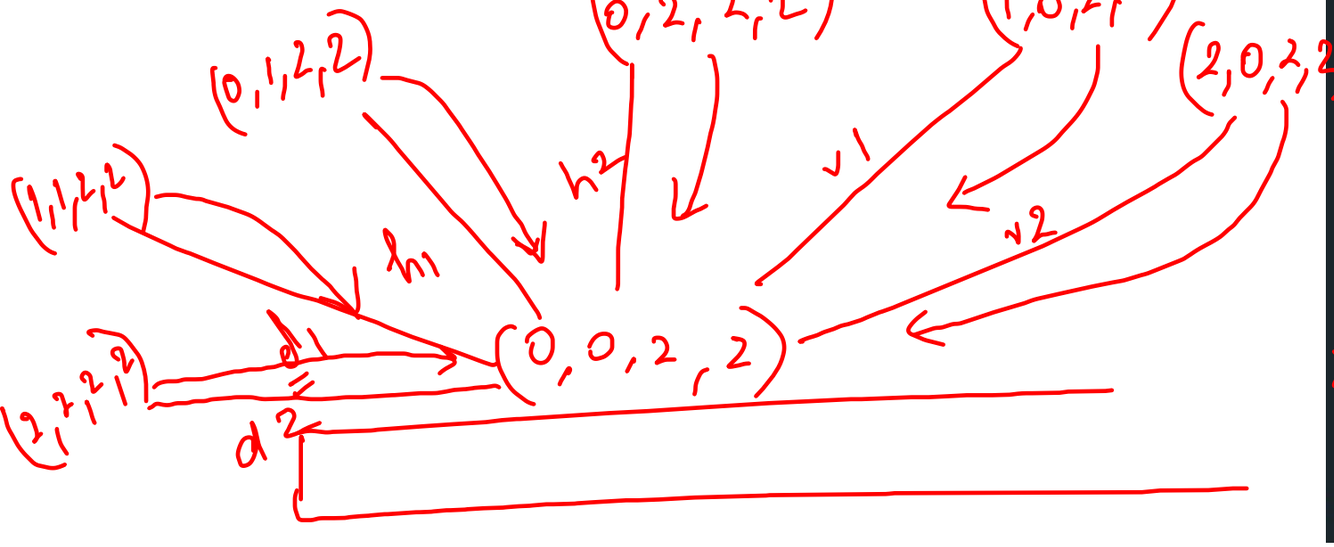


nr = 3

nc = 3

jmp = 2

```java
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"

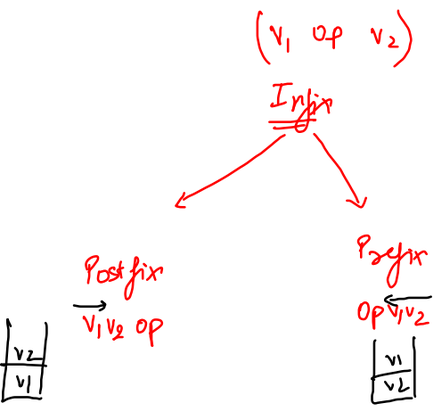
" " (4) = "o"

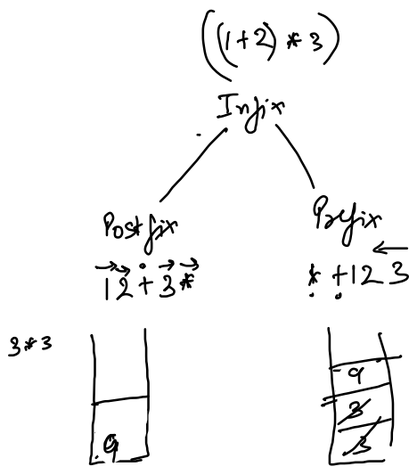
str.substring(5) = ""

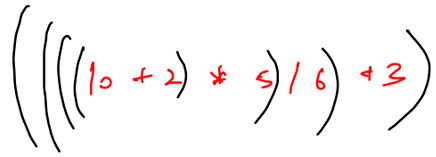
Special case

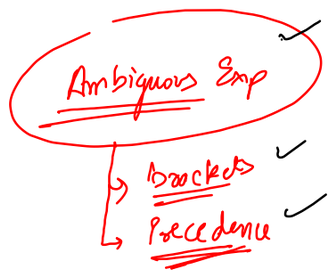
str.substring(idx);

[0 → length]









Post fix

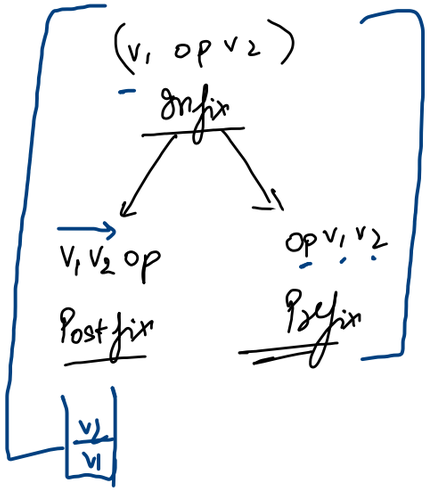
Prefix

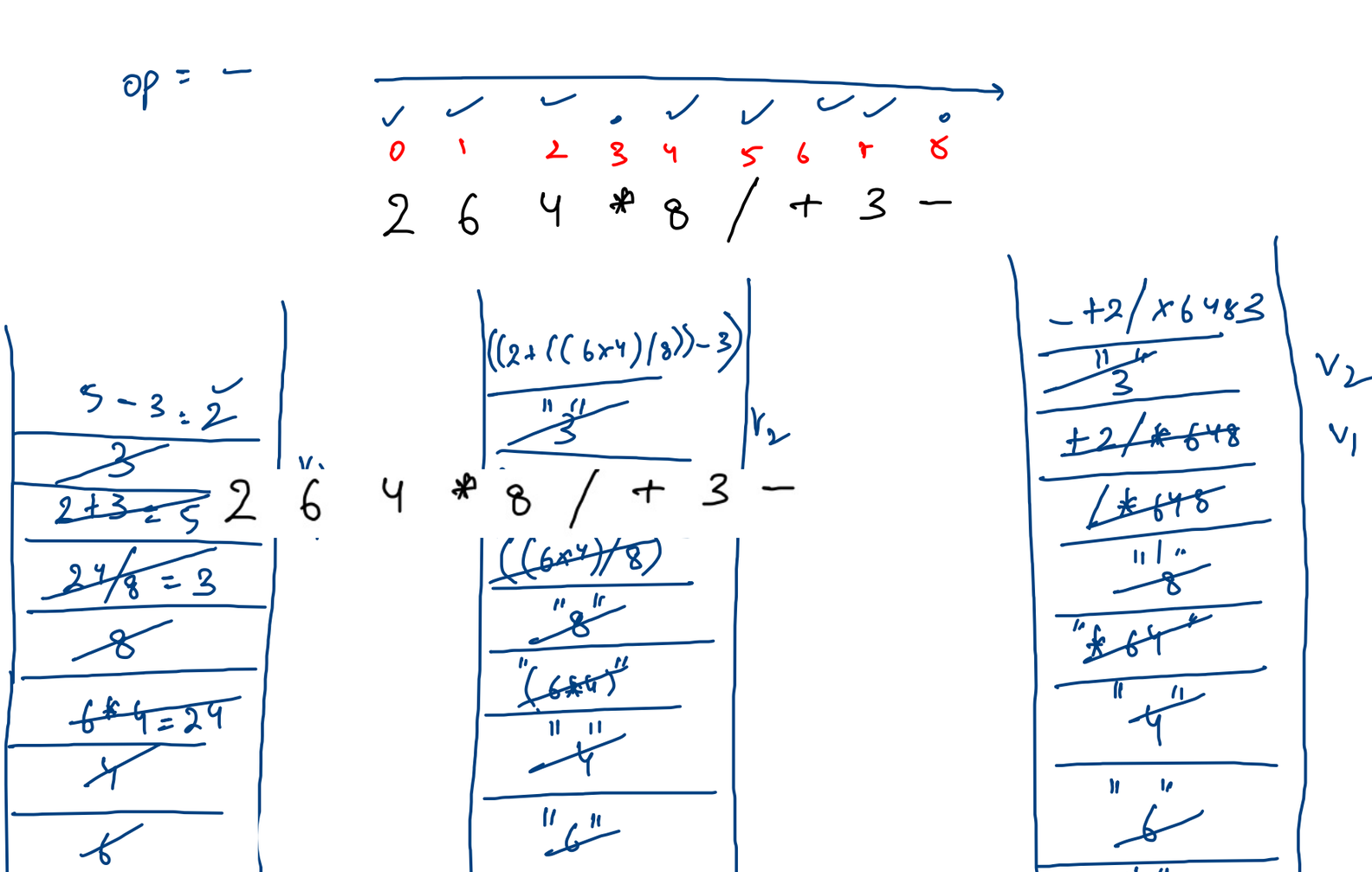
Sample Input

```
264*8/+3-
```

Sample Output

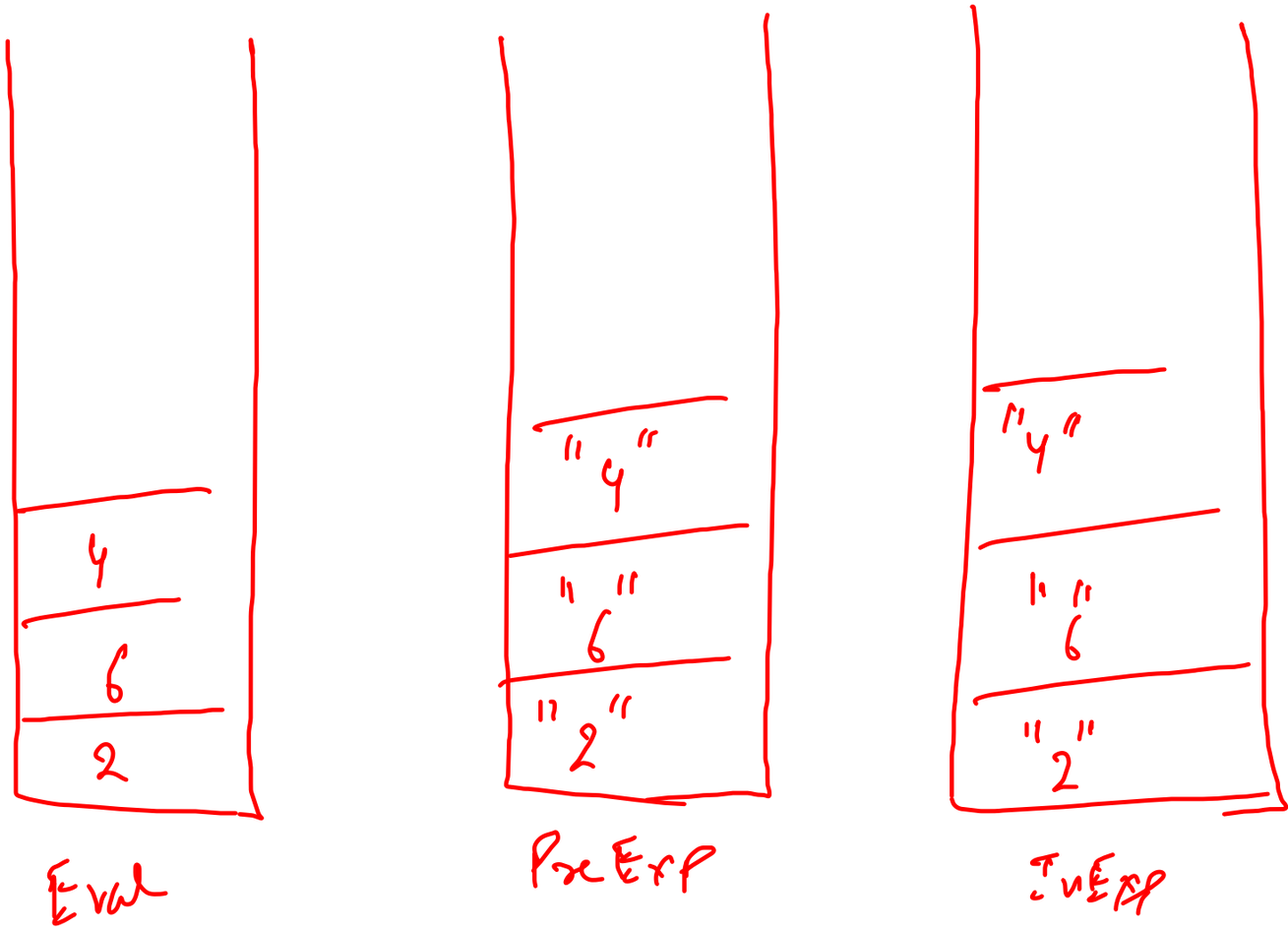
```
2
((2+((6*4)/8))-3)
-+2/*6483
```





2 6 4 * 8 / + 3 −

```java
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){ }
```

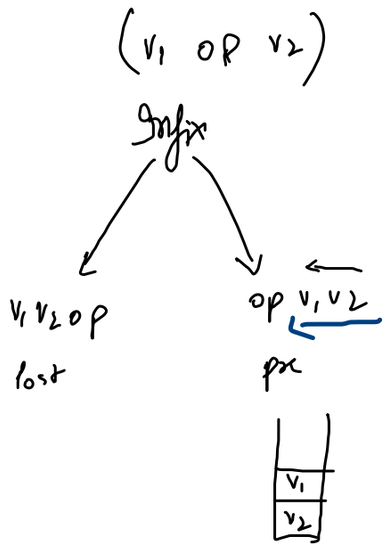


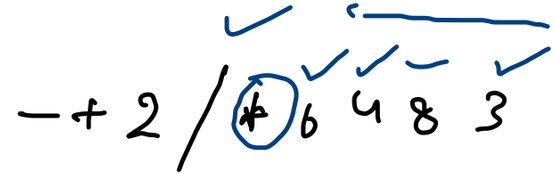
Sample Input

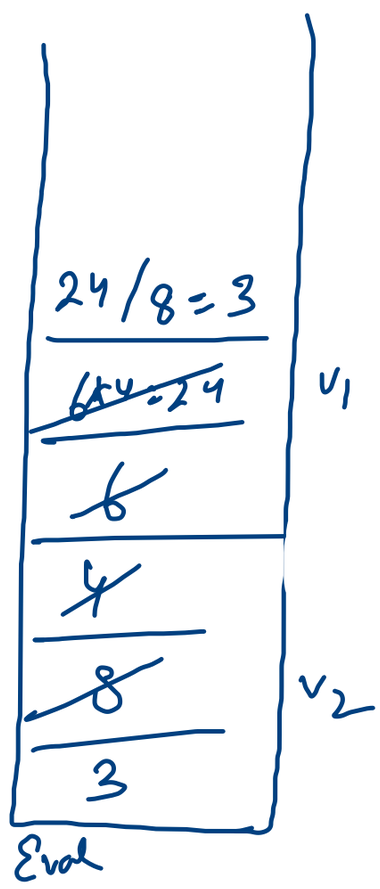
```
-+2/*6483
```

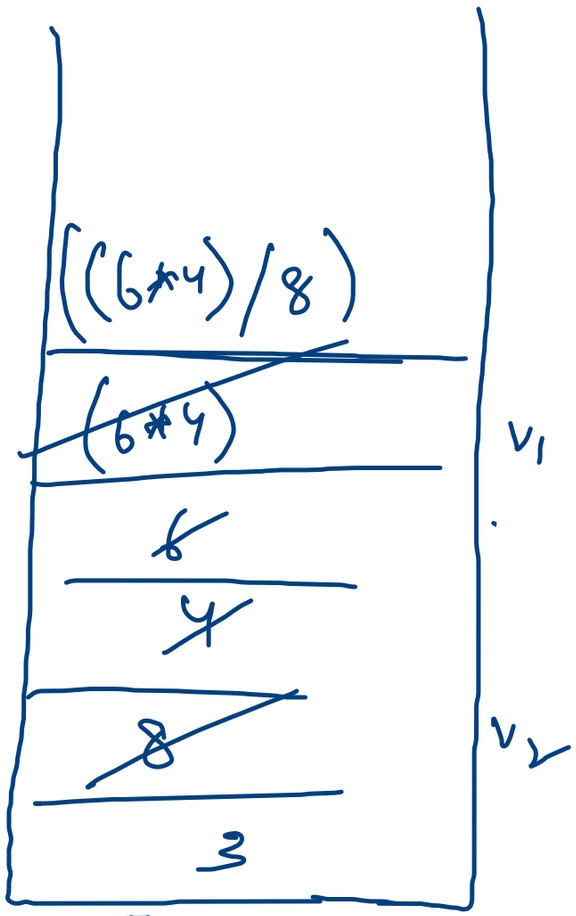
Sample Output

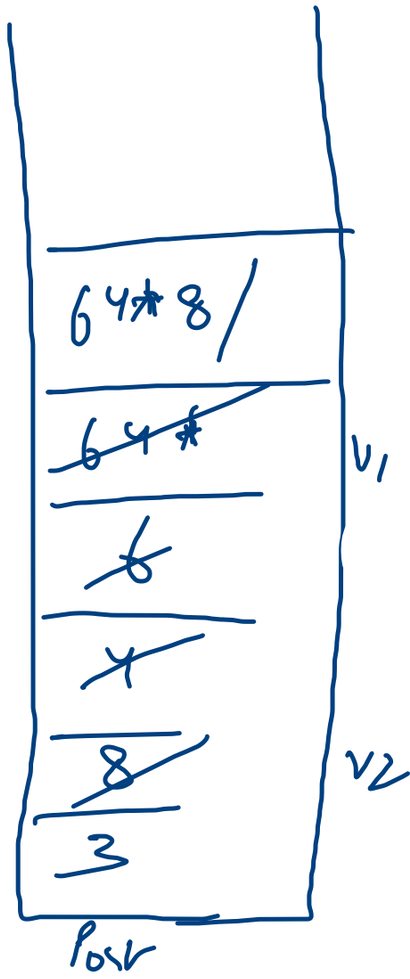
```
2
((2+((6*4)/8))-3)
264*8/+3-
```



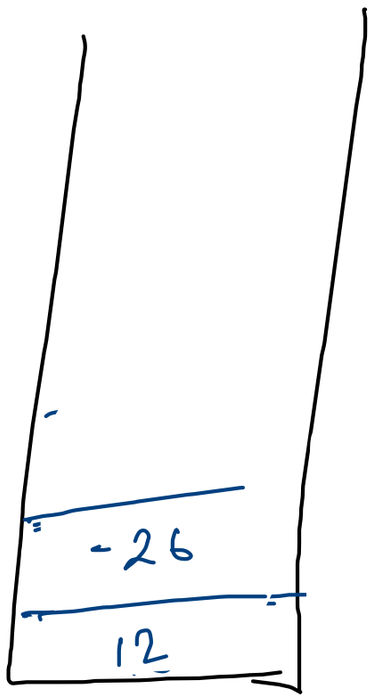


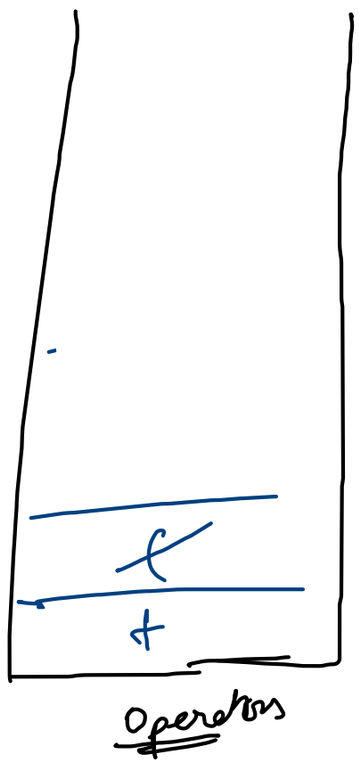






((8/2)*2)+4+(6-8*4)/8

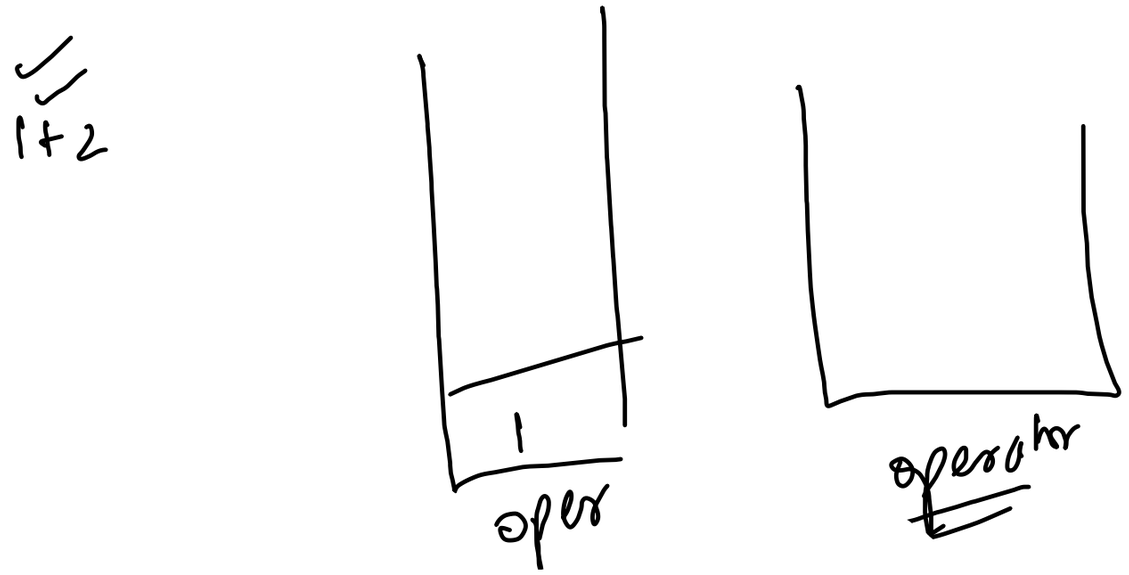




op=

st.peek =

(precedence(st.peek()) >= precedence(op))




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