Problem Solving 1

#QuoteOfTheWeek

"He who conquers himself is the mightiest warrior."

~ Confucius

@CSArunachal

01. B closest points to Origin

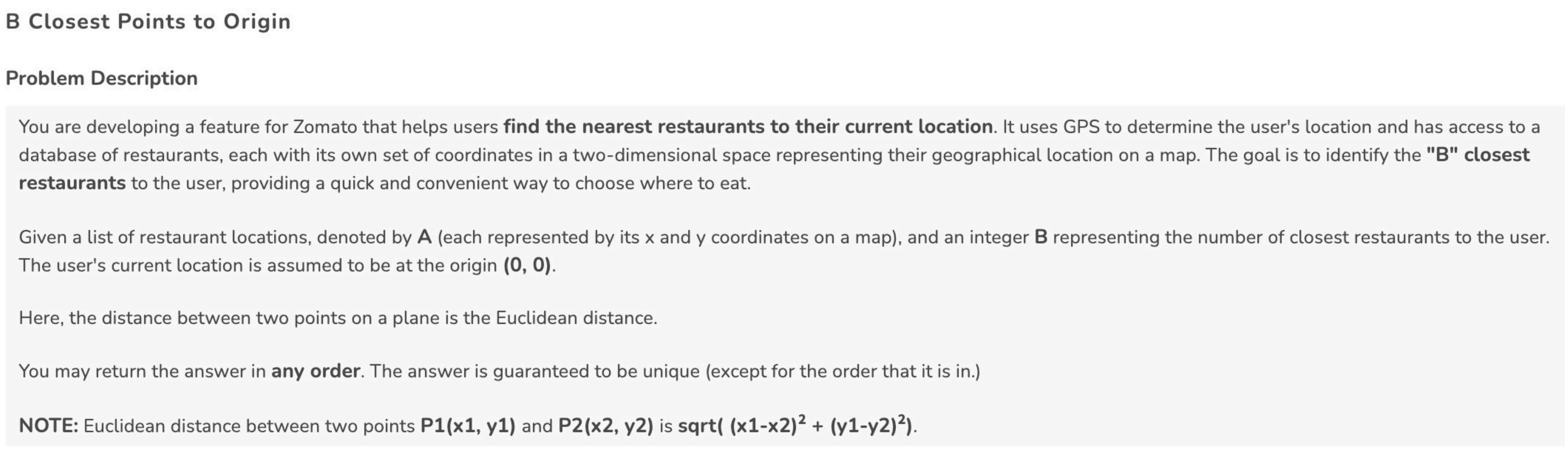
02. Allocate books

03. K reverse linkedlist

04. Infix to postfix

05. Check two bracket Expression

video solution



B Closest Points to Origin

Problem Description

You are developing a feature for Zomato that helps users **find the nearest restaurants to their current location**. It uses GPS to determine the user's location and has access to a database of restaurants, each with its own set of coordinates in a two-dimensional space representing their geographical location on a map. The goal is to identify the **"B" closest restaurants** to the user, providing a quick and convenient way to choose where to eat.

Given a list of restaurant locations, denoted by **A** (each represented by its x and y coordinates on a map), and an integer **B** representing the number of closest restaurants to the user. The user's current location is assumed to be at the origin **(0, 0)**.

Here, the distance between two points on a plane is the Euclidean distance.

You may return the answer in **any order**. The answer is guaranteed to be unique (except for the order that it is in.)

NOTE: Euclidean distance between two points **P1(x1, y1)** and **P2(x2, y2)** is **sqrt((x1-x2)^2 + (y1-y2)^2)**.

points [ ] [ ] = [[3, 3]
[5, -1]
[-2, 4]]

B = 2

user = [0, 0]

user

(x_1, y_1) —— (x_2, y_2)

$\Downarrow$

$$dis = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$(0,0)$ —— (x, y)

$\Downarrow$

$$dis = \sqrt{x^2 + y^2}$$

7:13 AM → 7:20 AM

points [ ][ ] = [[3, 3]
[5,-1]
[-2,4]]

k = 2

$$d_1 = \sqrt{3^2 + 3^2} = \sqrt{18}$$

$$d_2 = \sqrt{5^2 + (-1)^2} = \sqrt{26}$$

$$d_3 = \sqrt{(-2)^2 + 4^2} = \sqrt{20}$$

Ans [ ][ ] = [[3,3]
[-2,4]]

Brute force → Find distance of all points from origin & store it in the array.
Now, iterate on all those distances & get the B smallest points

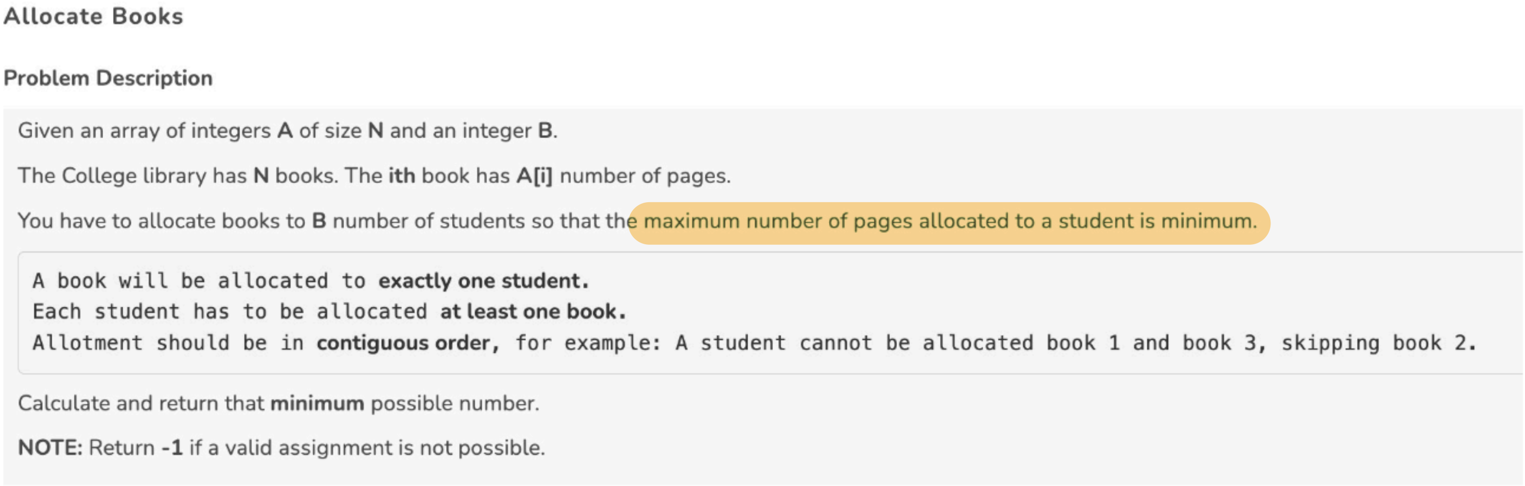
```
Arrays.sort ( points, new Comparator<>( ){
    public int compare ( int [] A, int [] B)
        int dis1 = A[0]*A[0] + A[1] * A[1]
        int dis2 = B[0]*B[0] + B[1]*B[1]
        if (dis1 < dis2) return -1;
        else if (dis1 > dis2) return 1;
        else return 0;
    }
}
```

Tc: O(nlogn)
Sc: O(n)

k = 2

points [ ][ ] = [[3, 3] [-2,4][5,-1]]

```
int [] ans = new int [k][2]

for ( i=0; i<k ; i++) {
|    ans [i] = points [i]
}
```



Allocate Books

Problem Description

Given an array of integers **A** of size **N** and an integer **B**.

The College library has **N** books. The **ith** book has **A[i]** number of pages.

You have to allocate books to **B** number of students so that the maximum number of pages allocated to a student is minimum.

```
A book will be allocated to exactly one student.
Each student has to be allocated at least one book.
Allotment should be in contiguous order, for example: A student cannot be allocated book 1 and book 3, skipping book 2.
```

Calculate and return that **minimum** possible number.

NOTE: Return **-1** if a valid assignment is not possible.

A[ ] = { 12 34 67 90 }   B = 2

S_1 =	12	46	113
S_2 =	191	157	90
	191	157	113

} Ans = 113

A[ ] = { 10 20 5 15 5 }   B = 2

S_1 = 1 book → 10

S_2 = 4 book → 45

} Max pages 45

S_1 = 2 book → 30

S_2 = 3 book → 25

} 30 → Ans = 30

S_1 = 3 books → 35

S_2 = 2 books → 20 } 35

S_1 = 4 books = 50

S_2 = 1 book = 5 } 50

7:44 → 7:54 pm

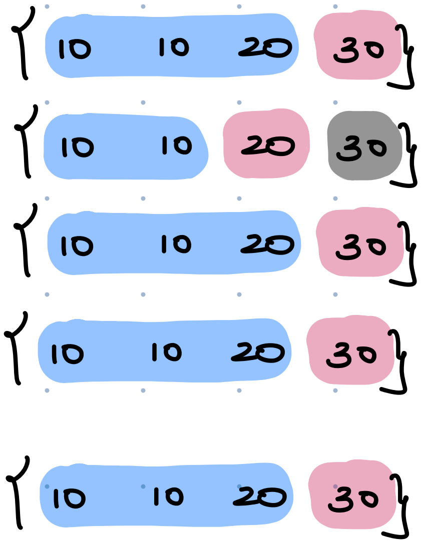
A[ ]= {10 10 20 30} B= 2

search space = { $\underbrace{30}_{\text{max of array}}^{\text{lo}}$ $\underbrace{70}_{\text{sum of array}}^{\text{hi}}$ }

lo	hi	mid	feasible to split mid no. of pages to B student	
30	70	50	{10 10 20 30}	Ans = 50

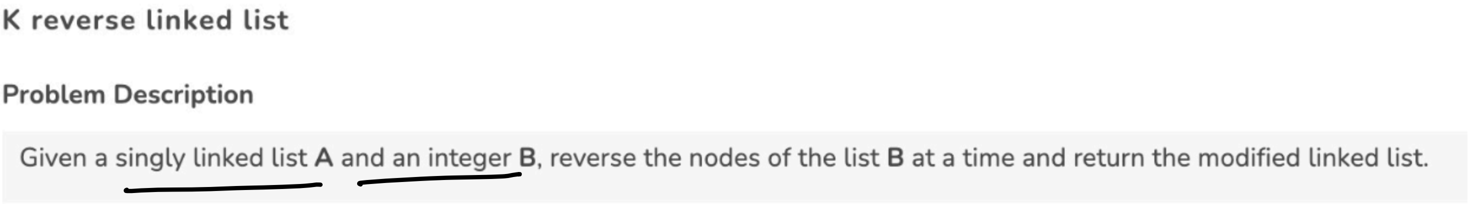
30	49	39	{10 10 20 30}	stu = 3 > B
				lo = mid + 1

40	49	44	{10 10 20 30}	Ans = 44
40	43	41	{10 10 20 30}	Ans = 41
40	40	40	{10 10 20 30}	Ans = 40
40	39	→	Stop	



```
int  minpages ( int [] A, int B)
    int lo = Max of Array (A)
    int hi = sum of Array (A)
    int ans = 0
    while (lo ≤ hi)
        mid = lo + (hi - lo)/2 ;
        if ( isfeasible ( A, B, mid))
            ans = mid;
            hi = mid - 1
        }
        else    lo = mid + 1
    }
    return ans;
}

boolean isfeasible ( int [] A, int B, int mid)

    int stu = 1 , sum = 0

    for ( i=0; i<n; i++)

        if ( sum + A[i] > mid )

            stu ++ ;

            sum = A[i];
        }
        else  sum += A[i];

    }
    return stu ≤ B;
}
```

8:14 Am → 8:24 AM



K reverse linked list

Problem Description

Given a singly linked list A and an integer B, reverse the nodes of the list B at a time and return the modified linked list.

head ↓

A[ ]= {1 2 3 4 5 6 7}

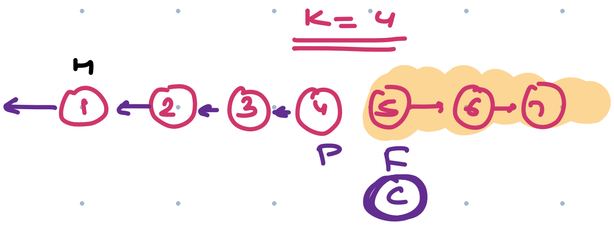
B = 2   2 → 1 → 4 → 3 → 6 → 5 → 7

B = 4   4 → 3 → 2 → 1 → 5 → 6 → 7

```
Node reversekgroup ( Node head, int k )
    int cnt = 0
    Node curr= head;
    while ( curr != Null && cnt < k)
        cnt ++;
        curr= curr. next;
    }

    if (count < k) return head;
    prev= Null forw = Null
    curr= head

    cnt = 0

    while ( cnt < k )
        forw = curr. next;
        curr. next = prev
        prev= curr ;  curr= forw;
        cnt ++;
    }
```

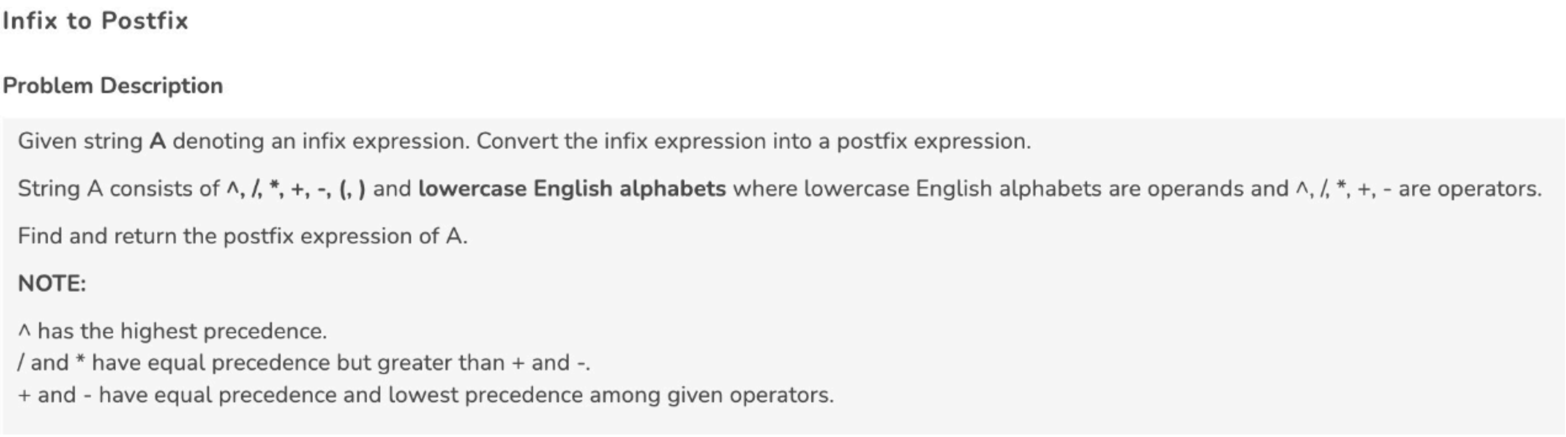
TC : O(N)

SC : O(N/k)



cnt = ∅ < 4
1 < 4
2 < 4
3 < 4
4 < 4 → stop

```
    head.next = reversekgroup(curr, k);
    return prev;
}
```



Infix to Postfix

Problem Description

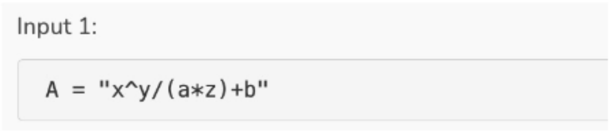
Given string **A** denoting an infix expression. Convert the infix expression into a postfix expression.

String A consists of ^, /, *, +, -, (,) and **lowercase English alphabets** where lowercase English alphabets are operands and ^, /, *, +, - are operators.

Find and return the postfix expression of A.

NOTE:

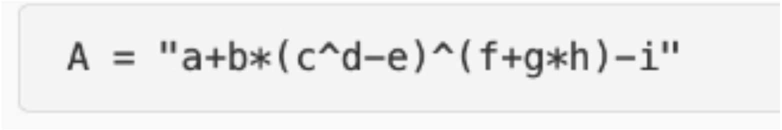
^ has the highest precedence.
/ and * have equal precedence but greater than + and -.
+ and - have equal precedence and lowest precedence among given operators.



Input 1:

```
A = "x^y/(a*z)+b"
```

```
x^y / (a*z) + b
x^y / az* + b
xy^ / az* + b
xy^ az* / + b
⇒ xy^ az* / b +
```



```
A = "a+b*(c^d-e)^(f+g*h)-i"
```

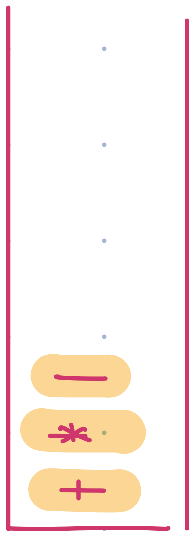
```
a+b*(c^d-e)^(f+g*h)-i
a+b*(cd^e-)^(f+gh*)-i
a+b* cd^e-fgh*+^ -i
a+b cd^e-fgh*+^* -i
ab cd^e-fgh*+^*+ -i
ab cd^e-fgh*+^*+i-
```

a - z ⇒ push it to AL

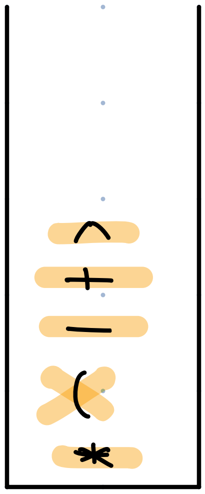
(⇒ push it to stack

) ⇒ resolve expression until opening bracket

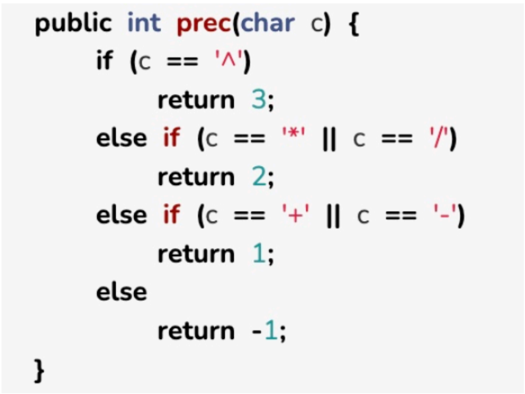
```
+ , * , - , /  ⇒  01. If st.empty() or ( is presed
                      on peek → st.push();

              else →
                      if (lower priority op is on peek)
                              ↳ st.push (curr operator)

                      else (higner priority op is presed)
                            or
                            equal
                                        ⇓
                                  resolve expression
```



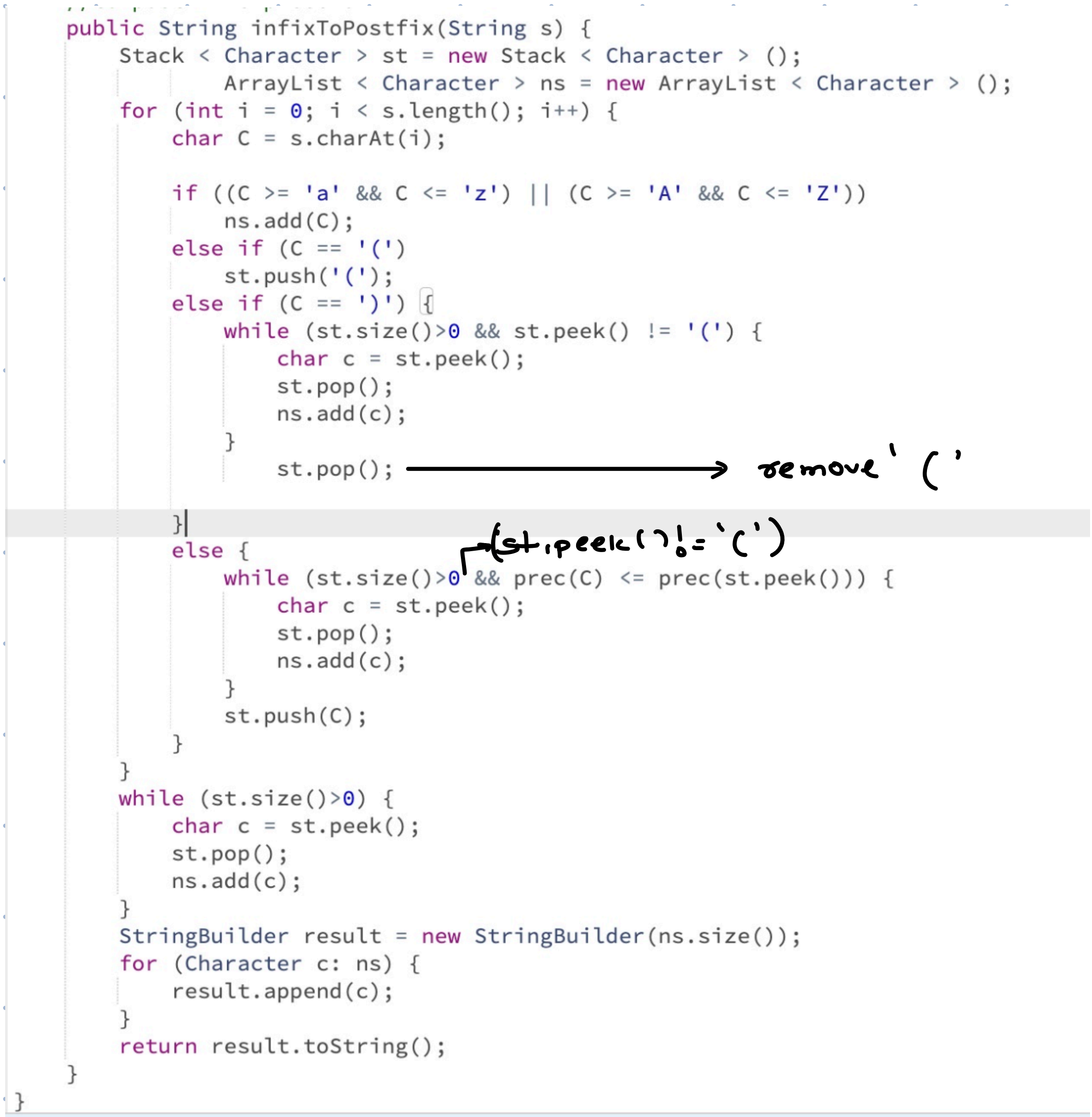
a + b * c - d ⇒ abc*+dstr= a * (b − c + d) ^ k



ans= a b c − d + k ^ *

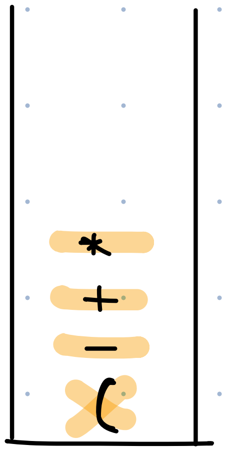


```
public int prec(char c) {
    if (c == '^')
        return 3;
    else if (c == '*' || c == '/')
        return 2;
    else if (c == '+' || c == '-')
        return 1;
    else
        return -1;
}
```



```
public String infixToPostfix(String s) {
    Stack < Character > st = new Stack < Character > ();
            ArrayList < Character > ns = new ArrayList < Character > ();
    for (int i = 0; i < s.length(); i++) {
        char C = s.charAt(i);

        if ((C >= 'a' && C <= 'z') || (C >= 'A' && C <= 'Z'))
            ns.add(C);
        else if (C == '(')
            st.push('(');
        else if (C == ')') {
            while (st.size()>0 && st.peek() != '(') {
                char c = st.peek();
                st.pop();
                ns.add(c);
            }
                st.pop();  ——————————→ remove '('

        }
        else {                  (st.peek()!='(')
            while (st.size()>0 && prec(C) <= prec(st.peek())) {
                char c = st.peek();
                st.pop();
                ns.add(c);
            }
            st.push(C);
        }
    }
    while (st.size()>0) {
        char c = st.peek();
        st.pop();
        ns.add(c);
    }
    StringBuilder result = new StringBuilder(ns.size());
    for (Character c: ns) {
        result.append(c);
    }
    return result.toString();
  }
}
```

(b − c + d * e)



bc − d e * +